2005

Learning ontology aware classifiers

Jun Zhang
Iowa State University

NOTE TO USERS

Learning ontology aware classifiers

by

Jun Zhang

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Computer Science

Program of Study Committee:
Vasant G. Honavar, Major Professor
Shashi K. Gadia
Dimitris Margaritis
Drena L. Dobbs
Hui-Hsien Chou

Iowa State University

Ames, Iowa

2005

Graduate College
Iowa State University

This is to certify that the doctoral dissertation of

Jun Zhang

has met the dissertation requirements of Iowa State University

Signature was redacted for privacy.

Major Professor

Signature was redacted for privacy.

For the Major Program

DEDICATION

To my family

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my thesis supervisor, Dr. Vasant Honavar, for his constant encouragement, mentoring, sustained support and help. I have benefited greatly from his extensive knowledge in machine learning research. He was always accessible for frequent discussions with me no matter how busy his schedule was. His many insightful comments and research ideas have inspired me to explore new directions in my research. My research endeavors would not have been successful without him.

My sincere appreciation goes to Dr. Shashi Gadia, Dr. Drena Dobbs, Dr. Dimitris Margaritis, and Dr. Hui-Hsien Chou. I would like to thank them for serving on my supervising committee, for reading my thesis and making constructive suggestions. I want to thank Dr. Dobbs and Dr. Gardia for taking the time to write recommendation letters for me.

Many thanks go to members in the Artificial Intelligence Research Lab for rewarding research discussions, and helpful comments on my thesis and numerous manuscripts. It is their friendship and camaraderie have made my years in Ames the most unforgettable ones in my life. I want to thank my close collaborators, Dae-Ki Kang, Feihong Wu, Doina Caragea, Jyotishman Pathak, Adrian Silvescu, for their contributions to my research. I would like to express my special thanks to Jihoon Yang and Rajesh Parekh for mentoring me regarding my career options. My thanks also goes to my friends, Haibo Luo, Tongjie Chen, Xin Lan, Wanhui Yang, Wei Huang, Dongping Xu, Haizheng Zhang, Changhui Yan, Jie Bao, Bo Li, Zhe Zhang, Zhuo Chen, Chao Su, Cien Shen, Duhong Chen, who made my days at Iowa State University enjoyable. I would like to thank Judy and Mallory Parmerlee for their friendship, and generous help in proofreading my thesis.

Finally, my special thanks are due to my wife, Min Huang, for her love and support throughout the years. I am also indebted to my parents for their unconditional love, support and encouragement.

ABSTRACT

Many applications of data-driven knowledge discovery processes call for the exploration of data from multiple points of view that reflect different ontological commitments on the part of the learner. This is particularly important in scientific applications in which specific ontological and representational commitments often reflect prior knowledge and working assumptions. Of particular interest in this context are algorithms for learning classifiers from ontologies and data. In many practical applications, it is often the case that the instances to be classified are specified using attributes at different level of precision, leading to partially specified instances. Against this background, my dissertation research is aimed at the design and analysis of algorithms for the construction of robust, compact, accurate and ontology-aware classifiers.

We have precisely formulated the problem of learning pattern classifiers from Attribute Value Taxonomies (i.e., AVT, a specific class of ontologies) and partially specified data. We have designed and implemented efficient algorithms for learning classifiers from such data sources. Based on a general strategy of top-down hypothesis refinement to search in a generalized hypothesis space, this framework can be extended to theoretically well-founded AVT-based variants of machine learning algorithms. Our AVT-guided learning algorithms adopt a general learning framework that takes into account the tradeoff between the complexity and the accuracy of the predictive models. This tradeoff enables us to learn a classifier that is both compact and accurate. We have systematically evaluated the resulting algorithms on machine learning benchmark data sets.

We have extended our approach to learning compact and accurate classifiers from semantically heterogeneous data sources. We presented a principled way to reduce the problem of learning from semantically heterogeneous data to the problem of learning from distributed par-

tially specified data by reconciling semantic heterogeneity using AVT-mappings, and described a sufficient statistics based solution. The resulting algorithm is exact, relative to its centralized counterpart, and our experimental results on synthesized distributed and semantically heterogeneous data verified our theoretical analysis on the exactness of the algorithm. We illustrated our approach to using this strategy to design AVT-guided algorithms for learning classifiers from semantically heterogeneous data using the Naïve Bayes classifier as an example. However, the proposed approach can be extended to a broad range of other machine learning algorithms.

CHAPTER 1. Introduction

This chapter provides an introduction to the main topics and the background of the thesis. A brief description of our approaches and the outline of the structure of the thesis is provided.

1.1 Background and Motivation

Rapid advancement of computing technology gives us the ability to store, manage and access vast amounts of data in a way far more than we could understand the data. Cheap computer hardware and fast communication devices have revolutionalized the way data is recorded and distributed. Tera bytes of data on credit card transactions, web based click-streams, geospatial data, human genome data and protein sequence data are collected, stored and updated every day.

However, as Cliff Stoll and Gary Schubert have pointed out philosophically, "Data is not information, Information is not knowledge, Knowledge is not understanding, Understanding is not wisdom". The growing amount of data is meaningless if we cannot find the right tool to extract useful information and knowledge from it.

Machine Learning (ML), along with Knowledge Discovery in Databases (KDD) and Data Mining (DM), provides us with theories and methodologies in exploring, analyzing, and eventually understanding the available massive amount of data.

Many data rich domains (e.g., biological sciences, space sciences, e-commerce) offer unprecedented opportunities in computer-assisted data driven knowledge acquisition in a number of applications, including in particular, data-driven scientific discovery in bioinformatics (e.g., characterization of protein sequence-structure-function relationships in computational molecular biology), environmental informatics, health informatics, data-driven decision making in

business and commerce, monitoring and control of complex systems (e.g., load forecasting in electric power networks), and security informatics (discovery of and countermeasures against frauds and attacks on critical information).

1.2 Machine Learning Methods in Data Driven Knowledge Discovery

Machine learning is the study of methods for programming computers to learn, and to improve their performance automatically with experience [Mitchell (1997); Dietterich (2003)]. A detailed understanding of information processing algorithms for machine learning leads to a better understanding of human learning abilities as well [Mitchell (1997); Duda et al. (2001)].

Due to the overwhelmingly large volume of data and the constantly changing nature of the phenomena in real world applications, it is almost impossible to extract useful information solely by hand. Machine learning algorithms help to build models from "training data", and the learned model can then be used to make predictions for new and unseen data. Machine learning has been established as an interdisciplinary subject which overlaps with artificial intelligence, statistics, knowledge discovery, data mining, and psychology. However, the emphasis of machine learning is on the accuracy and the effectiveness of the resulting learning system.

Machine learning in general, and supervised learning or learning from examples in particular, offers some of the most cost-effective approaches to automated or semi-automated knowledge acquisition (discovery of features, correlations, and other complex relationships and hypotheses that describe potentially interesting regularities from large data sets) in many data rich application domains. Examples of such applications include design of customizable information retrieval agents, synthesis of domain-specific information extraction engines, data-driven scientific discovery (e.g., characterization of protein sequence-structure-function relationships in computational molecular biology), discovery of patterns of coordinated attacks on distributed computer networks, data mining in business and commerce (e.g., credit risk assessment), automated diagnosis, and monitoring and control of complex systems (e.g., load forecasting in electric power networks). A large class of such applications involves synthesis of pattern classifiers using a training set of labeled instances (i.e., "training data"). Once trained,

such a classifier can be used to classify novel instances (i.e., instances not encountered during training). The conditions under which such classifiers can be expected to generalize well as well as methods for improving generalization have been the subject of extensive theoretical and experimental investigation.

Currently, there are a large variety of machine learning algorithms available, and they have been applied to a great number of real world problems. However, with new emerging demands of data rich applications, we have seen many new challenges for machine learning and knowledge discovery [Caragea et al. (2005a)]:

- In many real world applications, data repositories are large in size, dynamic, and physically distributed. Consequently, it is neither desirable nor feasible to gather all of the data in a centralized location for analysis. The ability of autonomous organizations to share raw data is limited due to a variety of reasons (e.g., privacy considerations). Thus, there is a growing need for efficient algorithms that can learn from distributed data sources without shipping large amount of data.

- In practice, data driven knowledge discovery occurs within a context. Autonomously developed data sources differ in terms of their underlying ontological commitments [Sowa (1999)], i.e., assumptions concerning the objects that exist in the world, the properties or attributes of the objects, the possible values of attributes, and their intended meaning, as well as the granularity or level of abstraction at which objects and their properties are described. In scientific discovery applications, because learners often need to examine data in different context from different perspectives, methods for context-dependent and ontology-aware information extraction from data with user-specified ontologies are needed.

- Collaborative scientific discovery applications often require users to be able to analyze data from multiple, semantically disparate data sources. For such data sources, there is no single privileged perspective that can serve all users, nor even a single user in every context. Hence, there is a need for methods that can dynamically and efficiently

extract and integrate information needed for learning (e.g., statistics) from distributed, semantically heterogeneous data based on user-specified ontologies and user-specified mappings between ontologies.

As a continually evolving research subject, machine learning has been applied to solve many other complex learning tasks, including supervised learning from sequence data, time series, spatial data, and other complex objects, and unsupervised learning for clustering, data visualization, density estimation, anomaly detection, and information retrieval, etc. [Mitchell (1997); Dietterich (2003)] We can be sure to see great advancement of the field of machine learning in the future.

1.3 Motivations for Learning from Ontologies and Data

The increasing need for information sharing between organizations, individuals and scientific communities has led to significant efforts to the construction of ontologies in Bioinformatics, E-commerce, Geospatial informatics, etc. Making ontological commitments (that are typically implicit in a data set) explicit enables users to explore data from different points of view, and at different levels of abstraction. Each point of view corresponds to a set of ontological (and representational) commitments regarding the domain of interest. The goal of incorporating ontologies in learning pattern classifiers is aimed at the design and analysis of algorithms for construction of robust, accurate and easy-to-comprehend classifiers from ontologies and data. Therefore, algorithms for learning from ontologies and data are of significant interests in practice. In the following, we listed several motivations for pursuing this research direction.

Our first motivation for exploring learning algorithms that can exploit user-supplied ontologies to analyze data stems from the significance of interaction between data and knowledge - in particular, knowledge that takes the form of ontology (or ontologies) used by the learner.

In many data-driven knowledge acquisition tasks, there is a need to explore data from multiple points of view that reflect different ontological commitments on the part of the learner. This is particularly important in scientific applications of machine learning where specific ontologi-

cal and representational commitments often reflect prior knowledge and working assumptions of scientists [Reinoso-Castillo et al. (2003); Sowa (1999)].

However, in such applications, there is no single universal ontology that can serve all users in every context or even a single user in every context. For example, in one context, the scientist may not consider it necessary to distinguish between different subfamilies of a family of proteins or different types of sequence patterns or structural features of proteins. In other cases, such distinctions may be desirable. Each ontological commitment that a user chooses is like an *ontological lens*. Because of different needs, different users might need different *lenses* regarding the domain of interests. Hence, methods for learning from ontologies and data are needed to support knowledge acquisition from data.

Ontologies also provide us with a rich repository of semantic information and relations. Use of ontology can enhance knowledge discovery and information extraction through standard machine learning methods. Moreover, ontologies help to facilitate information integration and data mining from semantically heterogeneous data sets. Hence, the exercise of designing algorithms for learning from ontologies and data can offer useful insights into the more general problems of machine learning.

Our second motivation for considering algorithms for learning from ontologies and data arises from the preference for comprehensible and simple, yet accurate and robust classifiers in many practical applications of data mining.

The availability of user-supplied ontologies presents the opportunity to learn classification rules that are expressed in terms of familiar hierarchically related concepts (or, abstract attribute values) leading to simpler, easier-to-comprehend rules [Zhang et al. (2002); Kohavi and Provost (2001)].

Taxonomies (also known as concept hierarchies) are among particularly common and useful classes of ontologies. A taxonomy is specified by a collection of names (types or concepts) and a set of type-subtype relations. Typically, attribute values are grouped into a tree structure, called attribute value taxonomies (AVTs), to reflect actual or assumed similarities among the attribute values in the domain of interest. When a taxonomy is defined over class labels (target

attribute), it is called a class taxonomy (CT).

There is a need to study strategies for transforming traditional inductive learning algorithms into ontology-guided (more specifically, AVT-guided and CT-guided) inductive learning algorithms for data-driven discovery of relationships at multiple levels of abstraction.

Our third motivation for considering ontology-guided learning algorithms arises from the need to learn from relatively small data sets where there is a greater chance of generating classifiers that overfit the training data.

A common approach used by statisticians when estimating from small samples involves shrinkage [McCallum et al. (1998); Duda et al. (2001)] or using abstract attribute values which correspond to sets of the original attribute values grouped according to an AVT (or abstract class labels from a class taxonomy) when there are too few instances that match any specific attribute value or class label to estimate the relevant statistics with adequate confidence.

Learning algorithms that exploit taxonomies can potentially perform shrinkage automatically thereby yielding robust classifiers. In other words, such algorithms can perform a built-in regularization or pruning to minimize over-fitting.

Our fourth motivation relates to the ability to learn from partially specified data when different instances are specified at different levels of precision relative to attribute value taxonomies

In many pattern classification tasks, it is often the case that the instances to be classified are specified at different levels of precision. That is, the value of a particular attribute, or the class label associated with an instance, or both are specified at different levels of precision in different instances, leading to *partially specified data* [Zhang and Honavar (2003); Zhang and Honavar (2004a)]. To illustrate this phenomenon, consider the AVT for the "color" attribute shown in Figure 1.1. We can use a precise shade of *Blue* for describing a blue object, such as *Sky Blue*, *Light Blue*, *Dark Blue*, and *Navy Blue*. But, in some cases when we lose this specificity, the object is simply described as *Blue* without specifying the precise shade of blue.

Partially specified instances are encountered quite often in practice. For example, in a medical diagnosis task, different cases may be described in terms of symptoms or results of

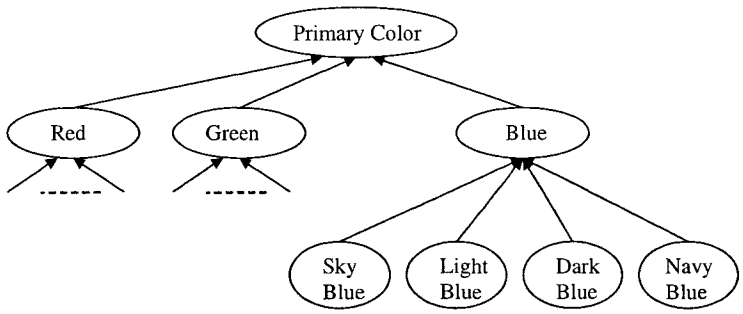


Figure 1.1 An Attribute Value Taxonomy for the Color attribute

diagnostic tests at different levels of precision. For example, a patient may be described as having cardiac arrhythmia without precise details of the type. While in another case, a patient has been diagnosed as having a detailed type of arrhythmia. In an intrusion detection task, the activity to be classified may be specified in terms of events described at different levels of precision. Likewise, the diagnosis (class label) associated with a set of symptoms may be specified at different levels of detail (e.g., breast cancer versus the precise type of breast cancer with respect to a class taxonomy). Partially specified data are also quite common in data mining applications in science (e.g., data driven approaches in functional genomics, characterization of macromolecular sequence-structure-function relationships), e-commerce (e.g., analysis of consumer behavior), security informatics (e.g., discovery of patterns of coordinated attacks on distributed computing and communications networks when network events, system calls, and user characteristics may be specified at different levels of detail), and social science.

Partially specified data are also unavoidable in knowledge acquisition scenarios which call for integration of information from semantically heterogeneous information sources [Reinoso-Castillo et al. (2003); Caragea et al. (2004a)]. Semantic differences between information sources arise as a direct consequence of differences in ontological commitments [Berners-Lee et al. (2001a)]. Hence, it is necessary for us to explore algorithms for learning from AVT and partially specified data.

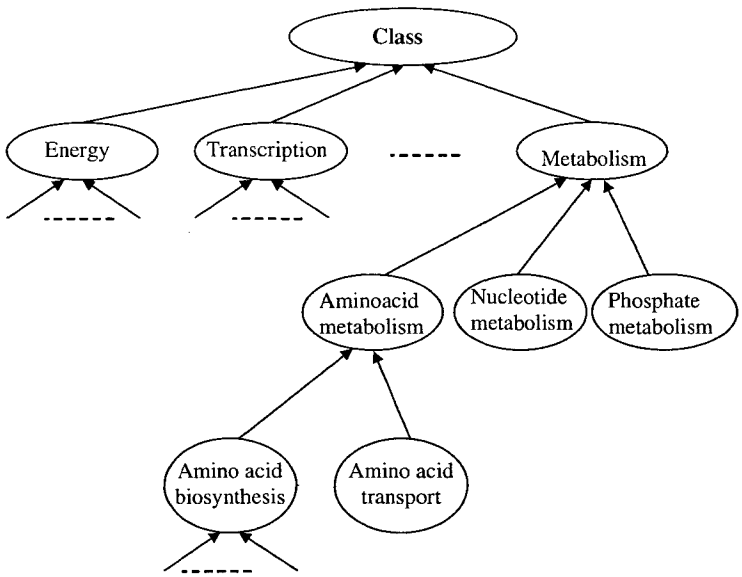


Figure 1.2 A partial Class Taxonomy for functional classification catalogue for S. Cerevisiae

1.4 Our Approaches

Against this background and the motivations, it is of significant practical interest to precisely formulate the problem of learning from ontologies (as a form of background knowledge or working assumptions) and data, and to explore the design space of algorithms for data-driven knowledge acquisition using *explicitly specified* ontologies (such as taxonomies).

In this thesis work, we formalize the problem of learning pattern classifiers from attribute value taxonomies and data (which can be partially specified), and propose AVT-guided learning algorithms. AVT-guided learning algorithms extend standard learning algorithms in principled ways so as to exploit the information provided by AVT. We have designed and implemented AVT-NBL [Zhang and Honavar (2004a); Zhang and Honavar (2004b)] and AVT-DTL [Zhang and Honavar (2003)] for learning AVT-guided Naive Bayes and Decision Tree classifiers, respectively. The standard Decision Trees or Naive Bayes learning algorithms can be viewed as special cases of AVT-DTL or AVT-NBL, where the AVT associated with each attribute has only one level. The root of such an AVT corresponds to the value of the attribute being unknown and the leaves correspond to the primitive values of the attribute.

Our general strategy is to implement a top-down AVT-guided search in the corresponding hypothesis space. An AVT-guided learning algorithm has a bias in favor of the level of abstraction based on more abstract attribute values (i.e., those that appear closer to the roots of the corresponding AVTs) that are sufficiently informative for classifying the training set. We start by building a classifier that is based on the most abstract level of abstraction and successively refine the classifier (hypothesis) by doing hypothesis refinement. Therefore, the learning algorithm generates a sequence of hypothesis refinements until a final optimal level of abstraction and an optimal classifier is obtained.

An AVT-guided learning algorithm adopts a general learning framework that takes into account the tradeoff between the complexity and the accuracy of the predictive models. For example, the Minimum Description Length (MDL) principle [Rissanen (1978)] is to compress the training data D and encode it by a hypothesis h such that it minimizes the length of the message that encodes both h and the data D given h. By making this tradeoff, we are able to learn classifiers that are both compact and accurate. We provide a general framework for learning classifiers from attribute value taxonomies and data. We illustrate the application of this framework in the case of AVT-based variants of decision tree and Naïve Bayes classifiers. However, this framework can be used to derive AVT-based variants of other learning algorithms, such as nonlinear regression classifiers, support vector machines, etc.

We extend our previous approach to learning compact and accurate classifier from partially specified semantically heterogeneous data sources. Our approach to AVT-guided learning from partially specified semantically heterogeneous data relies on our general strategy for transforming algorithms for learning from data into algorithms for learning from distributed, semantically heterogeneous data [Caragea et al. (2004a)]. This strategy is based on the decomposition of the learning task into an information extraction component (when *sufficient statistics* needed for learning are gathered) and a hypothesis generation component (that uses the *sufficient statistics* to generate or refine a current hypothesis).

We present a principled way to reduce the problem of learning from semantically heterogeneous data to the problem of learning from distributed partially specified data by reconciling

semantic heterogeneity using AVT-mappings, and describe a sufficient statistics based solution. The resulting algorithm is exact relative to its centralized counterpart, and our experimental results on synthesized distributed and semantically heterogeneous data verified our theoretical analysis on the exactness of the algorithm. We illustrate our approach to using this strategy to design AVT-guided algorithms for learning classifiers from semantically heterogeneous data using the Naive Bayes classifier as an example. However, the proposed approach can be extended to a broad range of machine learning algorithms including variants of Decision Tree, Bayesian networks, generalized linear models, and support vector machines.

1.5 Outline of the Thesis

The remainder of this thesis is organized as follows.

In Chapter 2, a brief introduction to learning classifiers from data is given. Preliminary concepts of ontologies and formal definitions of attribute value taxonomies and partially specified data are provided. We formally define the problem of learning classifiers from attribute value taxonomies and partially specified data, and define the problem of learning classifiers from distributed and semantically heterogeneous data sources. A brief survey of the related work on learning with missing values, learning classifiers from ontologies and data, and learning classifiers from distributed and heterogeneous data is given.

In Chapter 3, we present AVT-DTL, an AVT-guided decision tree learning algorithm, that is able to exploit user supplied attribute value taxonomies (AVTs) and to learn from partially specified data. We describe in detail how standard decision tree learning algorithms (DTL) can be extended in a principled way to AVT-DTL. We describe several alternative approaches to learning decision tree classifiers by data preprocessing. For performance evaluation, we present experimental results of AVT-DTL algorithm and make performance comparisons with standard decision tree learning algorithm and several alternative approaches.

In Chapter 4, we describe in detail AVT-NBL, a natural generalization of the Naïve Bayes learner (NBL), for learning classifiers from AVT and data, including partially specified data. We present experimental results by comparing AVT-NBL with standard Naïve Bayes Learner

(NBL) and an alternative approach (NBL on propositionalized data) on accuracy, complexity, and robustness of the induced classifiers.

In Chapter 5, we describe a general framework for designing algorithms to learn classifiers from AVT-extended data sources. Based on this framework, we demonstrate our approach by showing that AVT-DTL and AVT-NBL, presented in the previous two chapters, can be instantiated using this general framework.

In Chapter 6, we precisely define the problem of learning classifiers from distributed, semantically heterogeneous data sources viewed from a learner's perspective. We extend our previous approach for learning Naïve Bayes classifier from semantically homogeneous data with associated attribute value taxonomies and partially specified data to an approach for learning compact and accurate Naïve Bayes classifier from distributed and heterogeneous data sources. We reconcile semantic heterogeneity using AVT-mappings, and describe a sufficient statistics based solution to distributed data that is either horizontally fragmented or vertically fragmented. We also provide a theoretical analysis of the resulting classifier.

We conclude the thesis work in Chapter 7. A summary and the conclusions from my study are given. Some interesting future research problems are also addressed.

CHAPTER 2. Preliminary Concepts and Related Work

This chapter provides preliminaries and problem formulations for this thesis. First, we describe the basic concepts of learning classifiers from data. We provide preliminary concepts of ontologies and attribute value taxonomies. Then, we formalize the notions on partially specified data, and we formally define the problem of learning classifiers from attribute value taxonomies and partially specified data. We also define the problem of learning classifiers from distributed and semantically heterogeneous data sources. Finally, we examine related work in literature on learning with missing values, learning classifiers from ontologies and data, and learning classifiers from distributed and heterogeneous data.

2.1 Learning Pattern Classifiers from Data

Inductive learning algorithms offer a powerful approach to building classifiers for data-driven discovery of complex, apriori unknown relationships from data. The purpose of a pattern classifier is to label or categorize the data into a set of known classes. A classifier is constructed based on a set of labeled training data, and usually, but not always, an explicit set of classification rules can be extracted from the classifier, which provides us with a better understanding of the classification domain. Examples of classification of various domains include: (1) Loan eligibility for the applicants given the historical data pertaining to the customers, such as age, profession, income, family size, city location etc. A classifier is built to predict an applicant's ability to pay the monthly payment, or the likelihood of making such a payment; (2) Medical diagnosis based on different symptoms and/or lab test results for the patient; (3) Protein function prediction that assigns protein sequences into functional families based on characteristic motif compositions.

2.1.1 Data Format

As an essential part of a classification learning task, data can take different forms and formats. In the supervised learning scenario, each example (instance) is an ordered pair (X_p, c_{X_p}) where X_p is the input to the classifier and c_{X_p} is the class label. Typically, the set of class labels is assumed to be finite and the classes are assumed to be mutually exclusive. In many applications, the instance to be classified, that is, the input to the classifier is represented using a fixed set of features or attributes. The value of an attribute for a particular instance is a measurement of the quantity for that attribute. In this case, the p^{th} instance X_p is an ordered tuple of attribute values $(v_{1p}, v_{2p}, ..., v_{Np})$ where N is the number of attributes.

There are different kinds of attributes (so called variables in statistics) that are used to measure different aspects of the instance. Of particular interest to us in classification are *nominal* attribute, *numerical* attribute and *ordinal* attribute. Numerical attribute, sometimes called *continuous* attribute, assumes any value of a continuum of values, such as *Temperature* or *Weight*. A nominal attribute takes any value of a set of non-numerical values without particular order, such as *Weather Conditions* or a *Shopping List*, and is sometimes referred to as *categorical*. An ordinal attribute is similar to a nominal attribute, in which the values form a logical order, such as *Education Level* or *Job Ranking*. Other rarely used attribute types include *interval*, *ratio*, etc. An instance can be described using only one type of attribute or using a combination of different types of attributes. Occasionally, some attribute values may be missing in some of the instances to be classified. Consequently, techniques for dealing with missing attribute values during training, as well as classification, have received significant attention in the literature [Mitchell (1997); Duda et al. (2001)].

2.1.2 Instance Space and Hypothesis Class

A classifier is built on a set of labeled training instances. The whole data for training a classifier is called a sample, expressed as $D = \{(X_p, c_{X_p})|p = 1 \cdots |D|\}$. Normally, we assume that the sample is *independent and identically distributed* (iid), and all instances are drawn from the same unknown joint distribution $p(X, c)$. The order of the instances is not important.

We use instance space I to represent all possible instances defined throughout the domains of all attributes.

In pattern classification tasks, a hypothesis h is a classifier, represented in some hypothesis language or by a parameterized family of functions (e.g., decision trees, neural networks, Bayesian classifiers, SVM, etc.). A hypothesis class (or hypothesis space) H is a collection of all possible hypotheses using a particular hypothesis language or parameterized functional format. For example, a hypothesis space H for decision trees is the set of all possible decision trees.

2.1.3 Learning Classifiers from Data

Given the original training data set D, a hypothesis class H, and a performance criterion P, a learning algorithm L outputs a hypothesis $h \in H$ that optimizes P (e.g., finding a hypothesis that is most likely given the training data D, or finding a compact classifier with minimum classification error).

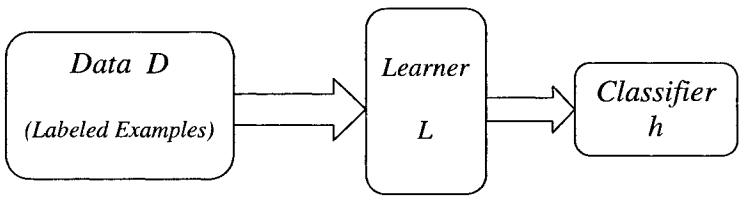


Figure 2.1 Learning Classifier from Data

Generally, we can use $g(X|\theta)$ to represent the classifier, where $g(\cdot)$ defines the hypothesis class, X is the input and θ are the parameters. We can define a loss function $R(\cdot)$ to compute the difference between the desired output and the output by classifier. Thus, we calculate the expected loss as the sum of losses over all instances [Alpaydin (2004)]:

$$E(\theta|D) = \sum_{p=1}^{|D|} R(c_{X_p}, g(X_p|\theta))$$

The learning algorithm uses an optimization procedure to find θ^* to minimize the approx-

imation error:

$$\theta^* = \underset{\theta}{\operatorname{argmin}} E(\theta|D)$$

Besides hypothesis language, different learning algorithms have different loss functions and different optimization procedures.

For many learning problems, data by itself is not sufficient to find a unique classifier, and we say such problems are ill-posed. Every learning algorithm has its own inductive bias, and does some model selection from the hypotheses that are consistent with the data. According to Dietterich (2003), there are triple trade-off factors in learning:

- The complexity of the hypothesis, or the capacity of the hypothesis class,
- The amount of training data,
- The generalization error on new examples.

In practice, we usually evaluate a classifier by doing *cross-validation* on the training data to help decide the best suitable classifier. Another approach to finding the optimal complexity of the classifier is by doing *regularization* [Breiman (1998)], which penalizes complex models in the augmented error function. *Minimum Description Length* (MLD) [Rissanen (1978)] makes the trade-off by using information theoretic measures on the *Kolmogorov complexity* of the data set.

2.2 Ontologies and Attribute Value Taxonomies

2.2.1 Ontologies

An ontology is an explicit specification of a conceptualization). A formal ontology is usually specified by a collection of names for concept and relationships organized in a partial ordering by the type-subtype relations. There are two kinds of ontology: generic ontology and domain specific ontology. The purpose of generic ontology is to include extensively categories of human knowledge. For example, CYC [Lenat (1995)], WordNet [Miller (1995)], and

Sensus [Swartout et al. (1996)] are generic ontologies. We are more interested in domain specific ontologies, because they are fine grained and directly related to a specific application domain. Such domain specific ontologies are usually constructed and maintained by a domain expert. There are many publicly available domain specific ontologies. For example, ontology for intrusion detection [Undercoffer and et al. (2004)], ontology for semantic web [Berners-Lee et al. (2001a)], ontology for e-commerce application [Kohavi and Provost (2001)], etc. Among the most popular ontologies for computational biology and bioinformatics is Gene Ontology [Ashburner and et al. (2000)], describing many aspects of macromolecular sequence, structure, and function. WHO has a Drug Dictionary [WHODD (2001)], and the US national library of medicine maintains a fairly large scale ontology called "Unified Medical Language System" [UMLS (2001)].

A full ontology itself is usually organized as a complex structure. Figure 2.2 shows only a small piece of Gene Ontology (GO). Some biological terms related to molecular functions, biological processes and cellular components are collected into a directed acyclic graph, where each node represents a term, and child-terms are either members or representatives or their parent-terms. GO is organized as a DAG (Directed Acyclic Graph). It has no cycles in the graph, but one child can have more than one parent.

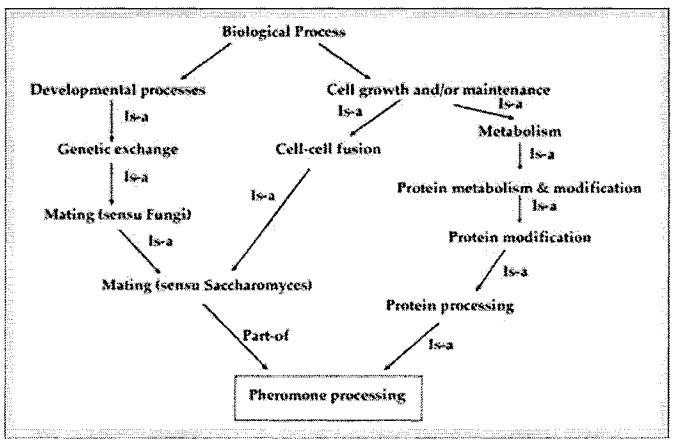


Figure 2.2 A piece of Gene Ontology

An ontological commitment defines the agreement to use the vocabulary in a coherent and

consistent manner within a specific data repository. Therefore, when data are distributed and collected by different entities, they may have different ontological commitments. Because ontologies can provide us with a rich repository of semantic information and rich relations, use of ontologies can enhance knowledge discovery and information extraction through standard machine learning methods.

There are commonly encountered types of ontologies that are of particular interest to learning, including taxonomies defined over attributes, taxonomies over class labels, and part-whole hierarchies over attributes. We will formally define attribute value taxonomy in next subsection.

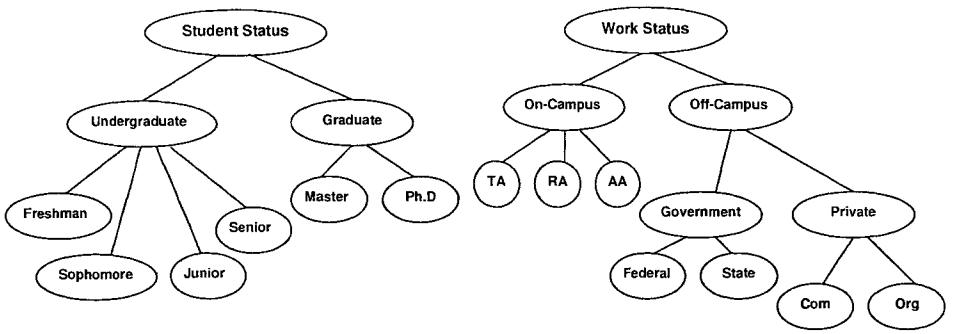
2.2.2 Attribute Value Taxonomies

Let $A = \{A_1, A_2, \ldots, A_N\}$ be an ordered set of nominal attributes, and let $dom(A_i)$ denote the set of values (the domain) of attribute A_i. We formally define attribute value taxonomy (AVT) as follows.

Definition 2.1 (ATTRIBUTE VALUE TAXONOMY) *Attribute value taxonomy $\mathcal{T}_i$ for attribute A_i is a tree structured concept hierarchy in the form of a partially order set $(dom(A_i), \prec)$, where $dom(A_i)$ is a finite set that enumerates all attribute values in A_i, and $\prec$ is the partial order that specifies* is-a *relationships among attribute values in $dom(A_i)$. Collectively, $T = \{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_N\}$ represents the ordered set of attribute value taxonomies associated with attributes $A_1, A_2, \ldots, A_N$.*

Let $Nodes(\mathcal{T}_i)$ represent the set of all values in $\mathcal{T}_i$, and $Root(\mathcal{T}_i)$ stand for the root of $\mathcal{T}_i$. The set of leaves of the tree, $Leaves(\mathcal{T}_i)$, corresponds to the set of *primitive values* of attribute A_i. The internal nodes of the tree (i.e., $Nodes(\mathcal{T}_i) - Leaves(\mathcal{T}_i)$) correspond to *abstract values* of attribute A_i. Each *arc* of the tree corresponds to a is-a relationship over attribute values in the AVT. Thus, an AVT defines an abstraction hierarchy over the values of an attribute.

For example, Figure 2.3 shows two attributes with corresponding AVTs for describing students in terms of their *student status* and *work status*, together with a sample data set collected by a university department based on the corresponding AVTs. With regard to the



Student ID	Student Status	Work Status	Hourly Income	Internship
60-421	Freshman	Org	$10/hr.	No
73-727	Master	Com	$30/hr.	Yes
81-253	Ph.D	RA	$20/hr.	No
75-455	Graduate	On-Campus	$20/hr.	No
32-719	Sophomore	AA	$15/hr.	No
42-139	Senior	Government	$25/hr.	Yes
66-338	Undergraduate	Federal	$25/hr.	Yes
.....	.....	.....	.....	.....

Figure 2.3 Two attribute value taxonomies on student status and work status and a sample data set based on the two corresponding AVTs.

AVT associated with *student status*, *Sophomore* is a primitive value while *Undergraduate* is an abstract value. *Undergraduate* is an abstraction of *Sophomore*, whereas *Sophomore* is a further specification of *Undergraduate*. We can similarly define AVT over ordered attributes as well as intervals defined over numerical attributes.

After Haussler (1988), we define a cut γ_i for $\mathcal{T}_i$ as follows. We also define a global cut Γ through T.

Definition 2.2 (CUT) *A cut γ_i is a subset of elements in $Nodes(\mathcal{T}_i)$ satisfying the following two properties:* (1) *For any leaf $m \in Leaves(\mathcal{T}_i)$, either $m \in \gamma_i$ or m is a descendant of an element $n \in \gamma_i$; and* (2) *For any two nodes $f, g \in \gamma_i$, f is neither a descendant nor an ancestor of g.*

Definition 2.3 (GLOBAL CUT) *Let Δ_i be the set of all valid cuts in $\mathcal{T}_i$ of attribute A_i, and $\Delta = \times_{i=1}^{N} \Delta_i$ be the cartesian product of the cuts through the individual AVTs. $\Gamma = \{\gamma_1, \gamma_2, \ldots, \gamma_N\}$ defines a global cut through $T = \{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_N\}$, where each $\gamma_i \in \Delta_i$ and $\Gamma \in \Delta$.*

The set of abstract attribute values at any given cut of $\mathcal{T}_i$ form a partition of the set of values at a lower level cut, and also induce a partition of all primitive values of A_i. For example in Figure 1, the cut {*Undergraduate*, *Graduate*} defines a partition over all the primitive values {*Freshman*, *Sophomore*, *Junior*, *Senior*, *Master*, *Ph.D*} in the *student status* attribute, and the cut {*On-Campus*, *Off-Campus*} defines a partition over its lower level cut {*On-Campus*, *Government*, *Private*} in the *work status* attribute.

Definition 2.4 (CUT REFINEMENT) *We say that a cut $\hat{\gamma}_i$ is a refinement of a cut γ_i if $\hat{\gamma}_i$ is obtained by replacing at least one attribute value $v \in \gamma_i$ by its descendants $\psi(v, \mathcal{T}_i)$. Conversely, γ_i is an abstraction of $\hat{\gamma}_i$. We say that a global cut $\hat{\Gamma}$ is a refinement of a global cut Γ if at least one cut in $\hat{\Gamma}$ is a refinement of a cut in Γ. Conversely, the global cut Γ is an abstraction of the global cut $\hat{\Gamma}$.*

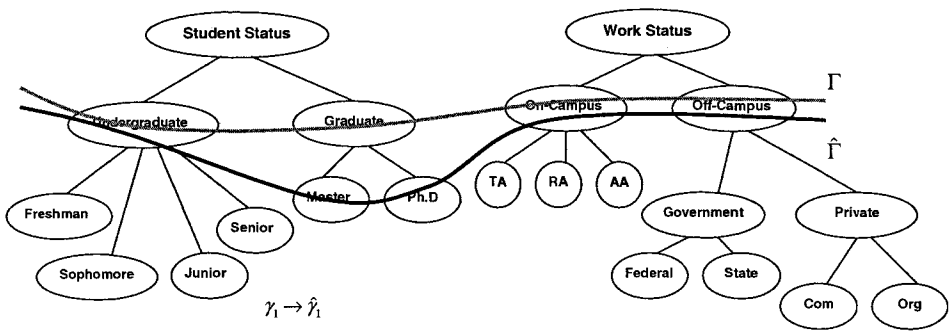


Figure 2.4 Cut refinement. The cut γ_1={*Undergraduate*, *Graduate*} in the *student status* attribute has been refined to $\hat{\gamma}_1$={*Undergraduate*, *Master*, *Ph.D*}, such that the global cut Γ={*Undergraduate*, *Graduate*, *On-Campus*, *Off-Campus*} has been refined to $\hat{\Gamma}$={*Undergraduate*, *Master*, *Ph.D*, *On-Campus*, *Off-Campus*}.

As an example, Figure 2.4 illustrates a demonstrative cut refinement process based on the AVTs shown in Figure 2.3. The cut γ_1={*Undergraduate*, *Graduate*} in the *student status*

attribute has been refined to $\hat{\gamma_1}$={*Undergraduate*, *Master*, *Ph.D*} by replacing "*Graduate*" with its two children "*Master*, *Ph.D*". Therefore, $\hat{\Gamma}$={*Undergraduate*, *Master*, *Ph.D*, *On-Campus*, *Off-Campus*} is a cut refinement of Γ={*Undergraduate*, *Graduate*,*On-Campus*, *Off-Campus*}.

2.3 Learning Classifiers from Attribute Value Taxonomies and Partially Specified Data

2.3.1 AVT Induced Abstract Instance Space

A classifier is built on a set of labeled training instances. The original instance space I without AVTs is an instance space defined over the domains of all attributes. We can formally define AVT-induced instance space as follows.

Definition 2.5 (ABSTRACT INSTANCE SPACE) *Any choice* Γ *of* $\Delta = \times_i \Delta_i$ *defines an abstract instance space* I_Γ. *When* $\exists i\, \gamma_i \in \Gamma$ *such that* $\gamma_i \neq Leaves(T_i)$, *the resulting instance space is an abstraction of the original instance space* I. *The original instance space is given by* $I = I_{\Gamma_0}$, *where* $\forall i\, \gamma_i \in \Gamma_0$, $\gamma_i = Values(A_i) = Leaves(T_i)$, *that is, the primitive values of the attributes* $A_1 \cdots A_N$.

Definition 2.6 (AVT-INDUCED INSTANCE SPACE) *A set of AVTs* $T = \{T_1 \cdots T_N\}$ *associated with a set of attributes* $A = \{A_1 \cdots A_N\}$ *induces an instance space* $I_T = \cup_{\Gamma \in \Delta} I_\Gamma$ *(the union of instance spaces induced by all of the the cuts through the set of AVTs* T*).*

2.3.2 Partially Specified Data

In order to facilitate a precise definition of partially specified data, we define two operations on AVT $\mathcal{T}_i$ associated with attribute A_i.

- $depth(\mathcal{T}_i, v(A_i))$ returns the length of the path from root to an attribute value $v(A_i)$ in the taxonomy;
- $leaf(\mathcal{T}_i, v(A_i))$ returns a Boolean value indicating if $v(A_i)$ is a leaf node in $\mathcal{T}_i$, that is if $v(A_i) \in Leaves(\mathcal{T}_i)$.

Definition 2.7 (PARTIALLY SPECIFIED DATA) *An instance X_p is represented by a tuple (v_{1p}, $v_{2p}, ..., v_{Np}$). X_p is:*

- *a partially specified instance if one or more of its attribute values are not primitive:* $\exists v_{ip} \in X_p$, $depth(\mathcal{T}_i, v_{ip}) \geq 0 \wedge \neg leaf(\mathcal{T}_i, v_{ip})$
- *a completely specified instance if* $\forall i\, v_{ip} \in Leaves(\mathcal{T}_i)$

Thus, a partially specified instance is an instance in which at least one of the attribute values is partially specified (or, *partially missing*). Relative to the AVT shown in Figure 2.3, the instance (*Ph.D*, *RA*) is a fully specified instance. The shaded instances in the sample data set are partially specified: (*Graduate*, *On-Campus*), (*Senior*, *Government*), (*Undergraduate*, *Federal*). The conventional missing value (normally recorded as "?") is a special case of partially specified attribute value, whose attribute value corresponds to the root of its AVT and contains no descriptive information about that attribute. We call such kind of missing *totally missing*, or *completely missing*.

Definition 2.8 (A PARTIALLY SPECIFIED DATA SET) *A partially specified data set D_T (relative to a set T of attribute value taxonomies) is a collection of instances drawn from I_T where each instance is labeled with the appropriate class label from $C = \{c_1, c_2, ..., c_M\}$, a finite set of mutually disjoint classes. Thus, $D_T \subseteq I_T \times C$.*

2.3.3 Learning Scenario

The problem of learning classifiers from AVT and partially specified data is a natural generalization of the problem of learning classifiers from data without AVT, and it is a special case for learning classifiers from ontology and data. In general, the typical hypothesis class H has been extended to H_O, where the original hypothesis language has been enriched by ontology O. The resulting hypothesis space H_O is a much larger space. In the case where the ontology is a set of attribute value taxonomies, the hypothesis space changes to H_T, a collection of hypothesis classes $\{H_\Gamma | \Gamma \in \Delta\}$. Each H_Γ corresponds to a hypothesis class with

regard to a global cut Γ in the AVTs. Because partial ordering exists among global cuts, it is obvious that the resulting hypothesis space H_T also has partial ordering structure.

Specifically, the problem of learning classifiers from AVT and data can be stated as follows:

Definition 2.9 (LEARNING CLASSIFIERS FROM AVT AND DATA) *Given a user-supplied set of AVTs T and a data set D_T of (possibly) partially specified labeled instances, construct a classifier $h_T : I_T \rightarrow C$ for assigning appropriate class labels to each instance in the instance space I_T.*

Of special interest are the cases in which the resulting hypothesis space H_T has structure that makes it possible to search it efficiently for a hypothesis h that is both concise as well as accurate.

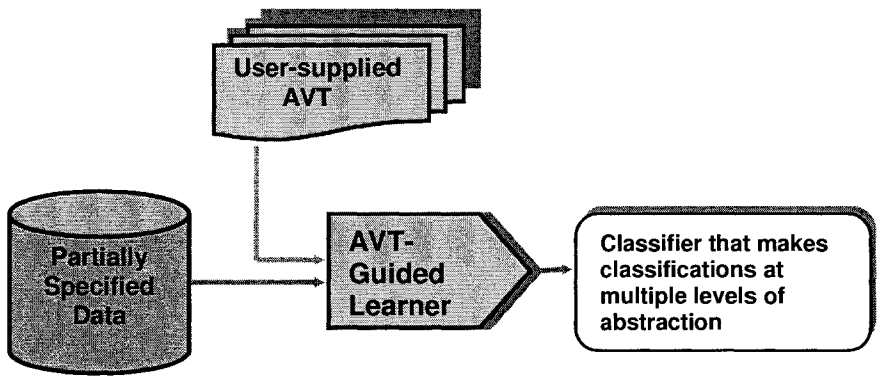


Figure 2.5 Learning classifiers from AVT and data.

Figure 2.5 shows a general picture of our learning scenario. Our learning algorithms take input partially specified data set (it may be a combination of fully specified and partially specified instances), and take input as well the user supplied AVTs. Our output is a pattern classifier that is able to generate classification rules at multiple levels of abstraction.

We describe AVT-DTL and AVT-NBL for learning AVT-guided Decision Tree and Naïve Bayes classifiers in details in chapters 3 and 4. We provide a general learning framework for AVT-based classifier learners in chapter 5.

2.4 Learning Classifiers from Distributed and Semantically Heterogeneous Data Sources

2.4.1 A Motivating Example

We use student information collected from different university departments as a motivating example to show the data sources from which we are going to learn a pattern classifier [Caragea et al. (2005b)].

Consider that two computer science departments from two universities independently collect information about their *Students* in connection to *Internships*. Suppose that the data D_1 collected by the first department is described by the attributes *Student ID*, *Student Status*, *Work Status*, *Monthly Income* and *Internship* and it is stored into a table as the one corresponding to D_1 in Table 2.1.

The data D_2 collected by the second department is described by the attributes *Univ. ID*, *Student Status*, *Work Status*, *Hourly Income* and *Internship* and it is stored into a table as the one corresponding to D_2 in Table 2.1.

Now that we have a specific user, who may come from either of the departments or from another university, we want to learn a classifier from the two data sources, but with his or her own perspective. Specifically we consider the case where the use has a different representative attributes, including *SSN*, *Student Status*, *Work Status*, *Yearly Income* and *Internship*. For example, the user may want to classify the instances (represented as in the entry corresponding to D_L in Table 2.1) into the target class *Internship* by using the classifier learned from the other two data sources.

To build classifiers from such different data sources requires learning algorithms have the ability to perform queries over the two data sources associated with the departments of interest from the user's perspective (e.g., *number of graduate students who did an internship*). However, we notice that the two data sources differ in terms of semantics from the user's perspective. In order to cope with this heterogeneity of semantics, the user must observe that the attributes *Student ID* in the first data source and *Univ. ID* in the second data source correspond to the

Table 2.1 Student data collected by two departments and a user data set

	Student ID	*Student Status*	*Work Status*	*Monthly Income*	*Internship*
D_1	44215	3rd year	Com	2530	yes
	49897	Master	On-Campus	1600	no
	23221	D.Phil.	RA	1800	no
	22178	M.Sc.	Private	3500	yes
	63255	1st year	Org	900	no
	Univ. ID	*Student Status*	*Work Status*	*Hourly Income*	*Internship*
D_2	223-11	Master	Com	34	yes
	219-65	Sophomore	AA	15	no
	223-98	Ph.D	On-Campus	18	no
	277-12	Senior	Off-Campus	28	yes
	290-33	Graduate	State	28	yes
	SSN	*Student Status*	*Work Status*	*Yearly Income*	*Internship*
D_L	-8475	Junior	AA	13000	
	-5287	Graduate	Private	38000	
	-7530	Undergraduate	Org	9000	

attribute *SSN* in the user data; the attributes *Monthly Income* and *Hourly Income* correspond to the attribute *Yearly Income*, etc.

In order to solve the semantic heterogeneity and establish the correspondence between values that two similar attributes can take, we need to associate types with attributes and to map the domain of the type of an attribute to the domain of the type of the corresponding attribute (e.g., *Hourly Income* to *Yearly Income* or *Student Status* in D_1 to *Student Status* in D_L). We can associate the type of an attribute with a simple hierarchical ontology. For example, in describing the attribute *student status*, Figure 6.1 shows two different AVTs associated with data source D_1 and D_2. Figure 6.2 shows a learner AVT for the same attribute. Examples of semantical correspondences in this case could be: $Post-Graduate$ in D_1 is equivalent to *Graduate* in D_L, *1st Year* in D_1 is equivalent to $Freshman$ in D_L, $Master$ in D_2 is hierarchically below *Graduate* in D_L, etc.

We also notice that data in different data sources could be described at different levels of abstraction. For instance, the attribute *Student Level* in D_1 is specified in a greater detail

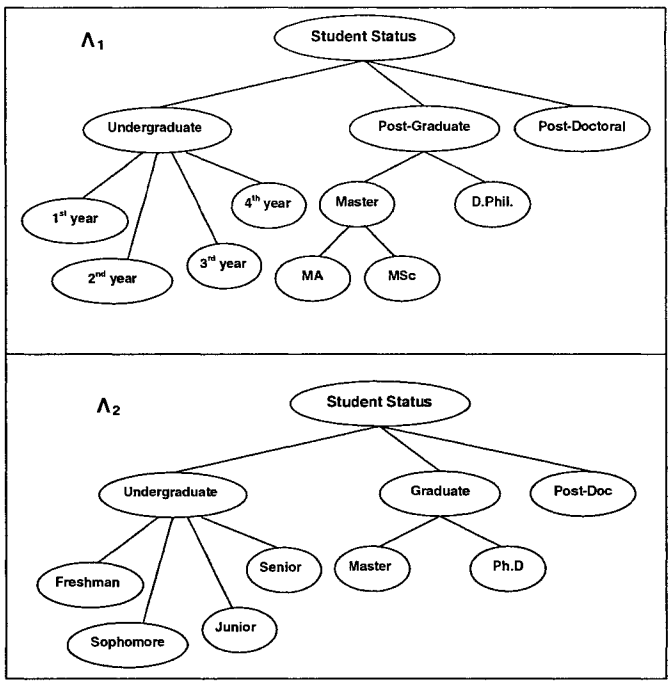


Figure 2.6 AVTs for data source D_1 and D_2 respectively

(lower level of abstraction) than the corresponding attribute *Student Program* in D_L. Instances within each data source can be specified at different levels of abstraction, and can be partially specified.

2.4.2 Distributed and Semantically Heterogeneous Data Source

Because data can be independently collected and operated, it is common that data for learning are physically distributed over several data sources. Here, we assume data D for learning are distributed over $D_1, D_2, \ldots, D_K$. Each data source D_i contains only a fragment of the whole data D, and we consider two common types of data fragmentation [Caragea et al. (2004b)]: *horizontal fragmentation*, wherein subsets of data tuples are stored at different locations; and *vertical fragmentation*, wherein sub-tuples of data tuples are stored at different locations.

The nature of distributed data sources imposes several constraints on learning classifiers from such data. One major constraint is the prohibition of shipping raw data from each of the

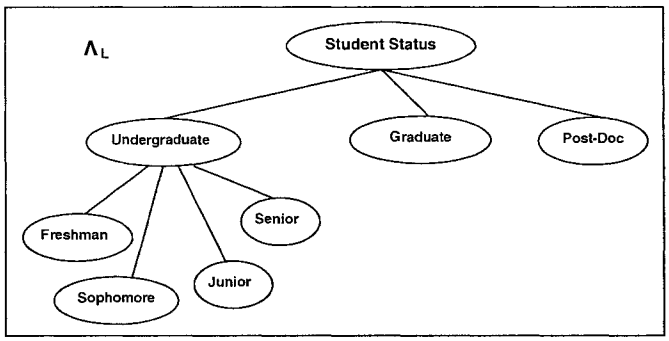


Figure 2.7 The Learner AVT

sites to a centralized location, but it allows collecting certain statistics from each site. When it is possible to obtain all necessary sufficient statistics from each and every individual site, learning classifiers from distributed data sources is possible without shipping the raw data.

In the motivating example, the student data are not only distributed in two different universities, but also they have different data semantics by following different ontologies. For example, they use different attribute value taxonomy for student status attribute. This leads to semantically heterogeneous data sources, where each data source D_i has an associated ontology O_i. We also assume that the learner has a learner's ontology O_L. The problem of learning from semantically heterogeneous data can be briefly defined as follows.

Definition 2.10 (Learning Classifiers from Semantically Heterogeneous Data) [Caragea et al. (2004a)] *Given the distributed, semantically heterogeneous data sources $D_1, \cdots, D_K$ with the associated ontologies $O_1, \cdots, O_K$ and a learner's ontology O_L, a hypothesis class H and a performance criterion P, the task of the learner L is to output a hypothesis $h \in H$ that optimizes P by integrating the data sources $D_1, \cdots, D_K$ according to the learner's ontology O_L.*

There are two different approaches in learning classifiers from semantically heterogeneous data. We can build a data warehouse for learning by getting all relevant data sources into one centralized location. However, this approach is sometimes not feasible nor desirable. A better approach is to perform necessary analysis on data where data and computational resources are available and only necessary information is transmitted to build a classifier. This enables a

learner to view the semantically heterogeneous data sources as though they were a collection of tables from the point of the learner's ontology. The key technique is to develop a mechanism to answering the statistical queries posed by the learner in terms of the learner's ontology from the heterogeneous data sources. We formally define *ontology-extended data sources*, and describe our solution to this learning problem in chapter 6.

2.5 Related Work

2.5.1 Learning in the Presence of Missing Values

There has been a long history in dealing with imperfect data in learning and statistical modelling. For real world scientific data exploration, the available data at the earlier stages of data mining are almost always imperfect. The early work of Bonnissone and Tong (1985) categorized three types of imperfect data: *incompleteness*, *imprecision*, and *uncertainty*. Incompleteness arises from the absence of a value, imprecision from the existence of a value which cannot be measured with suitable precision, and uncertainty from the fact that an agent has constructed a subjective opinion about the truth of a fact which it doesn't know for certain. A comprehensive survey is given by Parsons (1996).

Of special interest to machine learning and data mining is learning from data with missing values, which belongs to the category of *incompleteness*. Little and Rubin (1987) have defined three classes of missingness: (1) Observed at random (OAR); (2) Missing at random (MAR); (3) Missing completely at random (MCAR). There have been a variety of approaches to handling missing data. Most current machine learning approaches assume that data are MAR. The simplest technique for dealing with missing attribute values in training data is casewise deletion [Liu and White (1997)], thereby ignoring all the instances with missing values. One class of methods for handling missing values involves preprocessing techniques that fill in the missing attribute values. Commonly used preprocessing methods use statistical information of standard deviation [Pyle (1999)], or mean and mode [Han and Kamber (2001)] to replace the missing values, which are also called mean imputation. Other techniques for filling in missing attribute values may employ standard machine learning methods to predict the missing at-

tribute value based on the known attribute values, e.g., using the nearest neighbor estimator [Pyle (1999)] or decision trees [Quinlan (1986)]. C4.5, a popular decision tree construction algorithm [Quinlan (1993)] incorporates several methods to deal with missing values, which include: (1) replacing unknown values probabilistically according to distributions proportional to a known-value subset; (2) creating a new attribute value called unknown to branch instance with missing values; (3) breaking a missing value into fractional instances corresponding to estimated probabilities of observed attribute values, and deciding the most probable class label by exploring all possible branches during testing. White (1987) proposed a dynamic path generation method, in which only necessary path to classify the instances was generated. To deal with missing values, only attributes with known values were used to make a classification. When encountering missing values, it will choose the second most informative attribute. Similar to dynamic path generation, Friedman's lazy decision tree learning algorithm [Friedman et al. (1996)] is not restricted to a single decision tree, instead it creates a decision tree on the fly for each possible test instance. The Bayesian method has the built-in mechanism to deal with missing values. Some extensions on EM (Expectation-Maximization) algorithms were proposed to deal with missing attribute values [Heckerman (1999);Friedman (1997b)]. However, most of the above machine learning approaches to handling missing data are sophisticated and computationally expensive. This high computational cost prohibits them from scaling up to a larger data set.

There are very few papers in the database literature that focus on partially specified data. DeMichiel (1989) and Chen et al. (1996) proposed database models to handle imprecision using partial values and associated probabilities where a partial value refers to a set of possible values for an attribute. Aggregation operators were defined over partial values and were used to quantify the imprecision for the data with a higher level description in concept hierarchies, and aggregation equations minimize the Kullback-Leibler information divergence between the aggregated probability distribution and the data [McClean et al. (2001)].

None of the above approaches address the problem of learning from partially specified data, which is a generalization of learning from missing data in the presence of attribute value

taxonomies.

2.5.2 Learning Classifiers from Ontologies and Data

There is some work being done in the machine learning community on the problem of learning classifiers from attribute value taxonomies (sometimes called tree-structured attributes) and fully specified data in the case of decision trees and rules. Núñez (1991) outlined an approach to using ISA hierarchies in decision tree learning to minimize misclassification costs. Quinlan (1993) mentions handling of tree-structured attributes as a desirable extension to C4.5 decision tree package and suggests introducing nominal attributes for each level of the hierarchy and encoding examples using these new attributes. However, the Quinlan's encoding can only deal with feature values at fixed and balanced levels, because it lacks the ability to explore multiple levels of abstraction. Quinlan's C4.5 [Quinlan (1993)] provides an option called "subsetting", which allows C4.5 to consider splits based on subsets of attribute values (as opposed to single value) along each branch.

Almuallim et al. (1995) propose a technique for choosing a node in an AVT for a binary split using the information gain criterion. Almuallim et al. (1996) consider multiple split tests where each test corresponds to a cut through AVT. Because the number of cuts and hence the number of tests to be considered grows exponentially in the number of leaves of the hierarchy, this method scales poorly with the size of the hierarchy.

Taylor et al. (1997) proposed an algorithm to use ParkaDB as a tool to integrate ontologies and databases, and employ an evaluation function in decision tree learning to select the attribute-value pair to determine which level of concepts to include in a rule. However, the choice of evaluation function is so ad hoc and totally based on experience, such that it is difficult to generalize. Kudoh et al. (2003) proposed the *Information Theoretical Abstraction* (ITA) method to make data abstractions on attributes and to build decision tree classifiers based on the generated abstract attribute values. Their approach made no changes on decision tree learning algorithm, and because the abstraction is done only at one level, their algorithm is unable to explore a hierarchy of abstract attribute values at multiple levels of

abstraction. Kolluri and Metzler (1999) extended the *Spreading Activation Learning* (SAL) method to deal with numerical attributes in a simplified form of ISA taxonomies. desJardins et al. (2000) examined the use of feature hierarchies during Bayesian network learning, and proposed *Abstraction-Based Search* (ABS) to learn networks with compact structure and fewer parameters, resulting in better generalization performance. Chen et al. (2002) explore the use of concept hierarchies associated with attributes in databases to probabilistic relational models (PRMs) using a scoring-based searching algorithm to search for appropriate concept hierarchies to best fit the data.

Dhar and Tuzhilin (1993) and Hendler et al. (1996) describe the use of AVT in rule learning. Han and Fu (1996) proposed a method for exploring hierarchically structured background knowledge for learning association rules at multiple levels of abstraction. Cohen (1996a) has also incorporated set-valued attributes in the RIPPER algorithm for rule learning. However, set-valued attributes are not constrained by an AVT.

There is a large body of work on the use of *domain theories* to guide learning. The use of prior knowledge or domain theories specified typically in first order logic to guide learning from data in the ML-SMART system [Bergadano and Giordana (1990)]; the FOCL system [Pazzani and Kibler (1992)]; and the KBANN system which initializes a neural network using a domain theory specified in propositional logic [Towell and Shavlik (1994)]. AVT can be viewed as a restricted class of domain theories. Aronis and Provost (1996) used background knowledge to generate relational features for knowledge discovery. Aronis and Provost (1997) applied *breadth-first marker propagation* (BFMP) to exploit background knowledge in rule learning. However, the work on exploiting domain theories in learning has not focused on the effective use of AVT to learn classifiers from partially specified data. Walker (1980) was the first one to apply the taxonomies as background knowledge to information retrieval from large databases.

There has been some work on the use of class taxonomy (CT) in the learning of classifiers in scenarios where class labels correspond to nodes in a predefined class hierarchy. Clare and King (2001) have proposed a revised entropy calculation for constructing decision trees for assigning

protein sequences to hierarchically structured functional classes. Koller and Sahami (1997) describe the use of taxonomies over class labels to improve the performance of text classifiers. Karalic and Pirnat (1991) describe a technique that combines the outputs of separately learned binary classifiers to perform multi-label classification. McCallum et al. (1998) describes a Bayesian approach to multi-label learning for text documents that are represented by a mixture model which was trained by EM algorithm. Schapire and Singer (2000) describe extending the AdaBoost approach to handle multiple labels by producing and ranking possible classes for each document, preferring appropriate classes at the top of the ranking. Blockeel et al. (2002) use a clustering tree induction algorithm and a measure of suitable distances in hierarchy to do hierarchical multi-classification. Wu et al. (2005) defined the structured label classification problem, and proposed two learning algorithms, "Binarized Structured Label Learning" and "Split-based Structured Label Learning", to learn classifiers from data with structured labels in the bioinformatics domain. None of them, however, address the problem of learning from partially specified data (where class labels and/or attribute values are partially specified).

Automated construction of hierarchical taxonomies over attribute values and class labels is beginning to receive attention in the machine learning community. Examples include distributional clustering [Pereira et al. (1993)], extended FOCL and statistical clustering [Yamazaki et al. (1995)], information bottleneck [Slonim and Tishby (2000)], link-based clustering on relational data [Bhattacharya and Getoor (2004)]. Kang et al. (2004) implemented AVT-Learner, a Hierarchical Agglomerative Clustering algorithm to construct AVTs for learning. Joo et al. (2004) proposed an approach to automatically generate an AVT using a genetic algorithm. Such algorithms provide a source of AVT in domains where none are available. However, the work on exploiting domain theories in learning has not focused on the effective use of AVT to learn classifiers from partially specified data.

The problem of learning from ontologies and data has not been explored in its full generality, especially with regard to the problem of learning from partially specified data, in relation to a broad class of learning algorithms (e.g., Decision Trees, Naïve Bayes, Bayesian Networks, Support Vector Machines, etc.). Against this background, our research in this thesis is aimed

at the design, analysis, and application of algorithms for learning classifiers from partially specified data where the attribute values (or class labels) may be partially specified.

2.5.3 Learning Classifiers from Distributed, and Heterogeneous Data

Large scale exploratory analysis of distributed data is playing an increasingly important role in many data mining applications, and is becoming a newly emerging research field - Distributed Data Mining [Kargupta and Sivakumar (2004)]. A growing number of distributed data mining algorithms consider distributed data with heterogenous schemas defined by different sets of attributes across several data sites [Kargupta (2000); Strehl and Ghosh (2002); Caragea et al. (2000)]. Kargupta et al. (2001) proposed the collective principal component analysis (CPCA) algorithm to perform distributed PCA from heterogeneous sites. Forman and Zhang (2000) implemented a center-based distributed clustering algorithm that only requires the exchange of sufficient statistics. Jensen and Soparkar (2000) proposed an algorithm for learning associated rules from heterogeneous relational tables. For Bayesian Network learning from distributed data, Chen and Sivakumar (2002) and Chen et al. (2003) reported collective Bayesian Network learning algorithms for both structure and parameter learning.

Learning classifiers from distributed data has been an active research endeavor for the past several years. In the case of learning classifiers from homogeneous sites, the ensemble approaches have been applied to increase the classification accuracy by learning separate base classifiers from each data set and combining them using a weighted voting scheme. Domingos (1997) and Prodromidis et al. (2000) propose an *ensemble of classifiers* approach to learning from horizontally fragmented distributed data. Cho and Wuthrich (2002) proposed a distributed rule-based classifier system by selecting rules learned from data *fragments*. Gorodetski et al. (2000) addressed distributed learning in data fusion system. Fan et al. (1999) discussed an AdaBoost based ensemble approach from distributed scenarios. Bhatnagar and Srinivasan Bhatnagar and Srinivasan (1997) proposed an algorithm for learning decision tree classifiers from vertically fragmented distributed data. Kargupta et al. Kargupta et al. (1999) also described an algorithm for learning decision trees from vertically fragmented distributed data

using *Fourier coefficients*, a technique proposed by Mansour [Mansour (1994)]. Caragea et al. (2004b) provided a general framework for the design of algorithms for learning classifiers (e.g., decision trees) from distributed data that is provably exact with respect to its centralized counterpart.

In the case of learning classifiers from heterogeneous sites, a major issue is how to integrate information from heterogeneous distributed data sources. In terms of related work on data integration, Davidson *et al.* Davidson et al. (2001) and Eckman (2003) survey alternative approaches to data integration. Most of the traditional information integration approaches use mediator programs to integrate heterogeneus data sources. Levy (2000) proposed an approach based on logic, which is theoretically well-founded, but it doesn't deal with type heterogeneity. McClean et al. (2002) and McClean et al. (2003) provide an approach to answering aggregate queries formulated in a user ontology from statistical databases, but their framework assumes that there exists metadata, in terms of mappings between ontologies in the system. Maluf and Wiederhold (1997) proposed an ontology algebra for merging of ontologies. Bonatti et al. (2003) proposed a model of ontology-extended relational algebra. INDUS [Reinoso-Castillo et al. (2003)], a federated query-centric information integration platform, offers the functionality to integrate information from multiple heterogeneous data sources and provides the results according to a user-supplied ontology by specifying semantic correspondences between ontologies. Based on the INDUS framework, Caragea et al. (2004a) proposed a framework for learning classifier from semantically heterogeneous data sources by appropriately answering the statistical queries posed by the learner in terms of the learner's ontology from the heterogeneous data sources with their respective data source ontologies.

There are still many new challenges for learning from distributed and semantically heterogeneous data, such as the privacy preserving issue, communication cost analysis and real world applications, etc. The focus of our research in this thesis is to design algorithms that can learn from such semantically heterogeneous data sources and are able to find the best predictive models according to criteria that take into account both the complexity and the accuracy of the model.

CHAPTER 3. AVT-based Decision Tree Learner

In this chapter, we start by reviewing standard decision tree learning algorithms and the implications of being a good and comprehensible decision tree classifier. We present AVT-DTL, an AVT-guided decision tree learning algorithm, that is able to exploit user supplied attribute value taxonomies (AVTs) and learn from partially specified data. We describe in detail how standard decision tree learning algorithms (DTL) can be extended in a principled way to AVT-DTL. We describe several alternative approaches to learning decision tree classifiers by data preprocessing. For performance evaluation, we present experimental results of AVT-DTL algorithm and make performance comparisons with standard decision tree learning algorithm and several alternative approaches. Finally, we conclude with a summary and a discussion.

3.1 Learning Decision Tree Classifier from Data

Decision tree, a tree-based method for classification, is among the most popular and easily comprehensible pattern classifiers. The study of tree-based classification techniques can be traced back to the early work from both the machine learning community and the statistics community. The original work on "Concept Learning Systems" (CLS) by Hunt et al. (1966) is the pioneering work on decision tree, which was later extended by Quinlan to ID3 (ID stands for *Interactive Dichotomizer*) algorithm [Quinlan (1979)]. ID3 was further developed into C4, C4.5 and C5 by Quinlan (1993) to handle noisy data and probabilistic class membership at leaf nodes, and to incorporate different pruning methods to overcome the problem of data overfitting. The early work in the field of statistics began with the CART system by Breiman et al. (1984) for performing classification and regression using binary trees. Decision tree is so widely used in decision making and customer relationship management (CRM) that

many statistical software packages, such as SPSS, SAS, have included procedures for building classifiers by learning binary trees and different variants of decision trees. Many other statistical methods, including significance testing, overfitting avoidance, and their applications have also been introduced into the study of decision tree classifiers.

From the point of view of machine learning, decision tree learning is a supervised classification learning approach. The induced tree is learned from a training data set. Once constructed, the decision tree can be used to classify novel instances (i.e., instances not encountered during tree construction), and is straightforward in converting the tree into a set of explicit classification rules (i.e., IF-THEN rules).

We can also consider decision tree-based classification technique as non-parametric discriminant analysis. At each leaf node (i.e., terminal node) of the decision tree, posterior probabilities of class membership are estimated and implicitly encoded in decision tree classifier based on the class distributions of the training data. A new test instance follows a trail of tests over the internal nodes and reaches a leaf node with class conditional probabilities. Then classification is made based on those probabilities (e.g., assigning the instance to a class with maximum probability). As explained by White and Liu (1993), there is also a close correspondence between a probabilistic classification tree and a logit (logistic regression) model. Each path in the induced classification tree corresponds to a specific hierarchical logit model. For describing the same data, the process of *conditioning* on a variable in the statistical representation is equivalent to the process of *branching* in the tree representation. According to the analysis made by Breiman et al. (1984), the decision boundaries of decision tree on numerical data are axis-parallel, that is, decision tree partition the instance space I into portions perpendicular to the attribute axes. This could be one major limitation of a decision tree classifier when decision boundaries can not be assumed to be axis-parallel.

Next, we briefly review decision tree learning algorithms. Since an exhaustive search over a complete hypothesis space of decision trees is computationally infeasible, standard decision learning algorithms, such as ID3 and C4.5, implement a divide and conquer strategy that build decision tree recursively in a top-down style. Starting from a root node, it greedily selects the

best attribute based on some information criteria to make decision branches and partition data. Among the many different information criteria, entropy and Gini index are most commonly used.

Given the training data $D = I \times C$, where $C = \{c1, c2, \cdots, c_M\}$ is a finite set of mutually disjoint classes, we can calculate class probability distribution p_j by relative frequency on D for each $c_j \in C$, and get $p = \{p_1, p_2, \cdots, p_M\}$. We define entropy as:

$$I(p)_{entropy} = -\sum_{j=1}^{M} p_j \log p_j \tag{3.1}$$

and Gini index as:

$$I(p)_{Gini} = \sum_{i \neq j} p_i p_j = 1 - \sum_{j=1}^{M} p_j^2 \tag{3.2}$$

Both entropy and Gini index are impurity measures. The value will be zero if the probability is concentrated on one class, and will reach its maximum when the probabilities are evenly distributed among all classes. Given an impurity measure, we need to decide which attribute to split on. For any attribute A_i in $A = \{A_1, A_2, \cdots, A_N\}$, we calculate the decrease in average impurity by making a split on this attribute, and choose the one that maximizes the decrease of impurity. For attribute A_i, we denote its K possible values as $\{a_i^1, a_i^2, \cdots, a_i^K\}$. Given current data set D, we denote D_k as the portion of D for which attribute A_i has value a_i^k. We calculate the decrease of impurity $dI(p|D)$ for choosing A_i as the splitting attribute by:

$$dI(p|D) = I(p|D) - \sum_{k=1}^{K} \frac{|D_k|}{|D|} I(p|D_k) \tag{3.3}$$

Here, $I(p|D)$ can be any valid impurity measure, including entropy and Gini index. When entropy measure is chosen for $I(p|D)$, $dI(p|D)$ is called *information gain.*

There are other splitting criteria that have been shown to be effective with different preferences [Buja and Lee (2001)]. For a comprehensive survey on families of splitting criteria, please refer to Shih's paper [Shih (1999)].

It has been shown by Ciampi et al. (1987) and Clark and Pregibon (1992) that a decision tree can be seen as a probability model for the training data. They proved that the entropy-base selection criterion is the same as maximizing the deviance for the tree probability model.

Thus, the process of building a decision tree using entropy impurity measurement is essentially making maximum likelihood estimation over the training data.

As we have mentioned, the construction of a decision tree is a recursive process. It will terminate and make a leaf node only when it sees a *pure set* where all the instances belong to the same class, or it is statistically insignificant to further split the data using any of the attributes.

Following is the pseudo-code for decision tree learning algorithm. Note that $Majority\_Class(D_i)$ simply returns the most dominant class label in data set D_i.

Decision Tree Learning Algorithm:
Decision-Tree-Builder(Examples D, Attributes A, Default Class L)

1. If decision tree is $NULL$ Then create a root node for decision tree, set all examples to D, and set $L = Majority\_Class(D)$
2. If D is empty or does not pass the statistical significance test on any available attribute Then assign the label L to the node.
   Else If every instance in D has the same class label c Then Return(Leaf-node with c label)
3. Calculate the best attribute A_j by impurity measurement in Eq. (3.3)
4. Partition the examples D using the attribute values in A_j
   For each value $v_i \in A_j$ Do
   D_i = subset of D with value v_i
   Construct subtree by calling $Decision\text{-}Tree\text{-}Builder(D_i, A, Majority\_Class(D_i))$
   Add a new branch with tag v_i and connect to its subtree.
   End
5. Return the Decision Tree

Figure 3.1 General Procedure of Decision Tree Learning Algorithm

As a simple example, we use the following small data set (shown in Table 3.1) to build a decision tree (Shown in Figure 3.2) that can correctly classify the instances of the data. Rules can be easily extracted from the decision tree in the form of conjunctions of conditions (expressed in attribute values): *If* (Item1=RegCoke) AND (Item2=RegDorito) *Then* Young;

If (Item1=RegPepsi) AND (Item2=RegFrito) *Then* Midle-aged; *If* (Item1=RegPepsi) AND (Item2=BBQFrito) *Then* Midle-aged; *If* (Item1=DietCoke) *Then* Young; etc. There are 10 leaves in this decision tree, which correspond to 10 *If-Then* rules.

Table 3.1 A sample data set from customer purchase database

Customer	Item1	Item 2	Class
1	Diet Coke	Ranch Dorito	Young
2	AB OJ	CD Cereal	Old
3	Reg Coke	Reg Dorito	Young
4	Reg Coke	SB Chips	Mid-Aged
5	Diet Coke	Nacho Dorito	Young
6	Diet Pepsi	BBQ Frito	Mid-Aged
7	Reg Pepsi	Reg Frito	Mid-Aged
8	Skim Milk	CD Cereal	Old
9	Reg Pepsi	BBQ Frito	Mid-Aged
10	CD OJ	Bread	Old
11	Reg Pepsi	Popcorn	Young
12	AB Egg Nog	CD Steak	Old

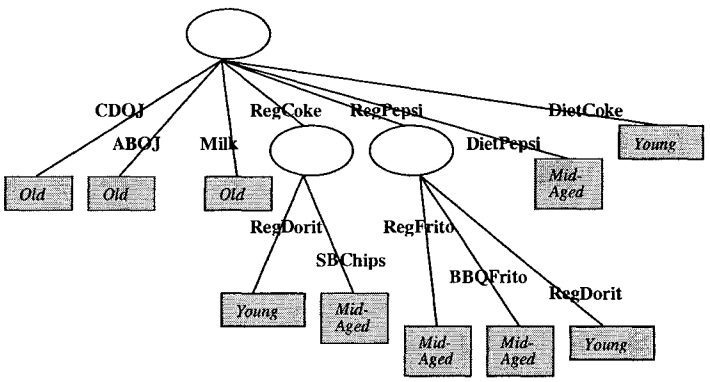


Figure 3.2 Decision Tree built on the customer purchase data set

When a decision tree classifier has been built, we would always want to know: (1)How accurate is the decision tree? (2)How does the decision tree generalize? (3)Is the decision tree the best tree that we can get? (4) How can we deal with missing attribute values?

For answering the first two questions, we can apply a standard evaluation procedure to test the performance of the tree that we have built. Some commonly used evaluation procedures include: using a separate test data set to test the accuracy, precision and recall of the decision tree classifier; applying *n-fold* cross validation and calculating confidence intervals; or choosing

other performance assessment methods [Cohen (1995)].

For answering the third question, we need to understand the meaning of "best" in the sense of a decision tree. According to Quinlan and Rivest (1989), a tree might be considered "best" if it has the smallest possible error rate when classifying previously unseen objects.

Even though it is sometimes possible to build a perfect decision tree (with 100% accuracy) for the training data, it may not be the preferred tree because it can perform very poorly in classifying unseen objects. This is the so called "overfitting" problem. We say a decision tree T overfits if there exists another tree T' that gives lower accuracy on the training data but higher accuracy on unseen test data. Overfitting is a general problem that all classifiers seek to minimize.

Because the real distribution that generates the data is unknown or does not even exist, there is no explicit procedure for a decision tree to minimize the error on unseen data. To overcome the problem of overfitting, several effective approaches have been proposed, including:

- Tree Pruning. There are basically two approaches in pruning decision tree. Pre-pruning stops growing the tree earlier according to statistical significance test (e.g., χ^2 test) in the middle of tree construction. Post-pruning allows growing a large tree first, and then iteratively pruning leaves off to minimize overfitting [Quinlan (1979); Breiman et al. (1984)].

- Regularization by MDL principle. According to Minimum Description Length principle, the best classifier is the one that requires the least number of bits to encode both the classifier and the data given the classifier [Rissanen (1978)]. The MDL principle has been applied as a regularization procedure for: (1) Choosing a splitting criterion [Ferri-Ramfrez et al. (2001)] in decision tree construction; (2) Inferring an optimal decision tree based on the encoding of the tree [Quinlan and Rivest (1989); Quinlan (1995)]; and (3) Pruning a decision tree based on the MDL principle [Mehta et al. (1995)].

As a matter of fact, all of the above approaches to minimizing overfitting are essentially making a tradeoff between the complexity and the accuracy of decision tree classifiers, which

also instantiate the general philosophy of "Occam's Razor". "Occam's Razor" states that *simpler explanations are more plausible* and any unnecessary complexity should be "shaved off".

As we have mentioned in Chapter 2, missing attribute values, or unknown attribute values, are unavoidable in real world data. Interestingly, it has been estimated that about half the code and about 80% of the programming effort in CART went into missing values [Friedman et al. (1996)]. There are different approaches in decision tree learning algorithms to handling missing values. As an example, in C4.5 Quinlan (1993), several methods to deal with missing values were proposed, including: (1) Replacing unknown values probabilistically according to distributions proportional to the known-value subset; (2) Creating a new attribute value called unknown to branch instance with missing values; (3) Breaking an instance with missing values into fractional instances corresponding to estimated probabilities of observed attribute values, and deciding the most probable class label by exploring all possible branches during testing.

Although it has been extensively studied, standard decision tree learning algorithm (DTL) is not equipped with the ability to exploit attribute value taxonomies, nor can it deal with partially specified data. Against this background, we propose AVT-DTL, a generalization of standard decision tree learning algorithms for building decision tree classifiers from attribute value taxonomies (AVT) and partially specified data.

3.2 AVT-DTL: Algorithm Description

We incorporate attribute value taxonomies into the learning of decision tree classifiers for the following considerations:

- An important goal of machine learning is to discover comprehensible, yet accurate and robust classifiers [Pazzani et al. (1997)]. The availability of AVT presents the opportunity to learn classification rules that are expressed in terms of *abstract* attribute values leading to simpler, accurate and easier-to-comprehend rules that are expressed using familiar hierarchically related concepts [Zhang et al. (2002) ;Kohavi and Provost (2001)].

- Exploiting AVT in learning classifier can potentially perform regularization to minimize overfitting when learning from relatively small data sets. A common approach used by statisticians when estimating from small samples involves *shrinkage* [McCallum et al. (1998)] to estimate the relevant statistics with adequate confidence. Learning algorithms that exploit AVT can potentially perform *shrinkage* automatically thereby yielding robust classifiers and minimizing over-fitting.

- The presence of explicitly defined AVT allows specification of data at different levels of precision, giving rise to *partially specified instances* [Zhang and Honavar (2003)]. The attribute value of a particular attribute can be specified at different levels of precision in different instances. For example, the medical diagnostic test results given by different institutions are presented at different levels of precision. Partially specified data are unavoidable in knowledge acquisition scenarios which call for integration of information from semantically heterogeneous information sources. Semantic differences between information sources arise as a direct consequence of differences in ontological commitments [Berners-Lee et al. (2001b)].

Hence, algorithms for learning decision tree classifiers from AVT and partially specified data are of significant practical interest. AVT-based Decision Tree Learner (AVT-DTL) [Zhang and Honavar (2003)] is a generalized version of standard decision tree learning algorithm for learning classifiers from Attribute Value Taxonomies (AVT) and data, and is able to deal with partially specified data. Next, we describe how standard decision tree learning algorithms (e.g., C4.5 [Quinlan (1993)]) can be extended in principled ways to exploit user-supplied AVT.

AVT-DTL accepts as input, user-supplied ordered set of AVTs $T = \{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_N\}$ corresponding to the attributes $\{A_1, A_2, \cdots A_N\}$ and a data set $D = \{(X_p, c_{X_p})\}$ of labelled examples, where $X_p \in \mathbf{I_A}$ is a partially or fully specified instance and $c_{X_p} \in \mathbf{C}$ is the corresponding class label. The task of the learner is to construct a decision tree classifier for assigning an instance X_p to one of several mutually exclusive classes.

AVT-DTL adopts the general principle of Minimum Description Length, and the preference is to favor shorter and more compact trees, which we believe will provide more general

rules which could summarize several lower level rules, and generalize well on new instances. Compared to standard decision tree learning algorithms, AVT-DTL has the following main characteristics:

- AVT-DTL implements a top-down AVT-guided search in decision tree hypothesis space.
- AVT-DTL has a bias in favor of splits based on more abstract attribute values (i.e., those that appear closer to the roots of the corresponding AVTs) that are sufficiently informative for classifying the training set.
- AVT-DTL chooses not just a particular attribute, but also an appropriate level of abstraction that corresponds to a cut through the AVT.

To facilitate description of AVT-DTL, we introduce the following notations, and for integrity, we reiterate some of the notions we already introduced in the preliminary section of chapter 2.

- Let $A = \{A_1, A_2, \ldots, A_N\}$ be an ordered sequence of attribute names. Let $T = \{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_N\}$ be the corresponding set of AVTs. Let $C = \{c_1, c_2, ..., c_M\}$ be a finite set of mutually disjoint classes.
- Let $\psi(v, \mathcal{T}_i)$ be the set of descendants of a node with value v in a taxonomy $\mathcal{T}_i$.
- Let $\pi(v, \mathcal{T}_i)$ be the set of all children (direct descendants) of a node with value v in $\mathcal{T}_i$.
- Let $\Lambda(v, \mathcal{T}_i)$ the list of ancestors, including the root, for v in $\mathcal{T}_i$.
- Let $\sigma_i(v, S)$ be the frequency count of value v of attribute A_i in a training set S.
- Let $Counts(\mathcal{T}_i)$ be a *tree of counts* corresponding to nodes in $\mathcal{T}_i$.
- Let $P_S = \{p_1^{(S)}, p_2^{(S)}, \cdots, p_n^{(S)}\}$ be a pointing vector (also called AVT frontier, corresponding to a global cut) for a set of instances S, where $p_i^{(S)}$ is a pointer to an attribute value in $\mathcal{T}_i$ of attribute A_i.
- Let $\Phi(P_S) = true$ if and only if $\forall p_i^{(S)} \in P_S$, and $\psi(p_i(S), \mathcal{T}_i) = \{\}$.

The AVT-DTL algorithm consists of two major steps: (a) Computation of the frequency counts given the user-supplied AVTs; (b) Construction of the decision tree based on the frequency counts estimations on AVTs.

3.2.1 Computation of Frequency Counts on AVT

Frequency counts on AVTs serve as important statistics that need to be estimated before conducting search in the decision tree hypothesis space. In building decision tree classifier, frequency counts provide the basis to calculate impurity measurement (information-based criterion) on the data set in deciding the best split, and they also serve as probabilistic distributions for breaking partially (or completely) missing attribute values into fractional values among the descendants and use the weighted instance values to compute their contribution of corresponding instances to entropy calculation for the split being considered.

Given an attribute value taxonomy $\mathcal{T}_i$ for attribute A_i, we can define a tree of frequency counts $Counts(\mathcal{T}_i)$, such that there is an one-to-one correspondence between the nodes of the AVT $\mathcal{T}_i$ and the nodes of the corresponding $Counts(\mathcal{T}_i)$. The procedure of calculating frequency counts $Counts(\mathcal{T}_i)$ is described in Figure 3.3.

We use a simple example to illustrate the estimation of class conditional frequency counts when some of the instances are partially specified. On the AVT for *student status* shown in Figure 3.4-(A), we mark each attribute value with a count showing the total number of positively labeled ("+") instances having that specific value. First, we aggregate the counts upwards from each node to its ancestors. For example, in Figure 3.4-(B), the four counts 10, 20, 5, 15 on primitive attribute values *Freshman*, *Sophomore*, *Junior*, and *Senior* add up to 50 as the count for *Undergraduate*. Since we also have 15 instances that are partially specified with the value *Undergraduate*, the two counts (15 and 50) aggregate again towards the root. Next, we distribute the counts of a partially specified attribute value downwards according to the distributions of values among their descendant nodes. For example, 15, the count of partially specified attribute value *Undergraduate*, is propagated down into fractional counts 3, 6, 1.5, 4.5 for *Freshman*, *Sophomore*, *Junior*, and *Senior* (See Figure 3.4-(C) for values in

1. Create a root node, set sample set to S, and set $P_S = \{A1, A2, \cdots, An\}$ so that elements of P_S point to the roots of the corresponding AVTs $\{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_N\}$.

2. Initialize frequency counts for AVT by scanning training samples:

   a) Accumulate the frequency counts associated with each value of each attribute based on the values that appear in instances in S. Thus, for each $\mathcal{T}_i \in T$, we compute $\sigma_i(v, S)$ associated with all attribute values v that correspond to nodes in $\mathcal{T}_i$.

   b) For each $\mathcal{T}_i \in T$, update the counts associated with ancestors of nodes in $\mathcal{T}_i$ which received non-zero counts as a result of step a) by propagating the counts up from each such node v to its ancestors. Let $Counts(\mathcal{T}_i)$ be the resulting counts.

   c) For each $\mathcal{T}_i \in T$, and each partially specified attribute value v in each instance $I_j \in S$,, calculate the fractional counts recursively for all descendants of v in T_i based on $Counts(\mathcal{T}_i)$ and update $Counts(\mathcal{T}_i)$. That is, for each d in $\psi(v, \mathcal{T}_i)$, update $\sigma_i(d, S)$ as follows:

$$\sigma_i(d,S) = \begin{cases} \left(\frac{\sigma_i(d,S)}{|\pi(v,\mathcal{T}_i)|}\right) & \text{If } \sum_{d\in\pi(v,\mathcal{T}_i)} \sigma_i(d,S) = 0, \\ \sigma_i(d,S)\left(1 + \frac{\sigma_i(v,S) - \sum_{d\in\pi(v,\mathcal{T}_i)} \sigma_i(d,S)}{\sum_{d\in\pi(v,\mathcal{T}_i)} \sigma_i(d,S)}\right) & \text{Otherwise.} \end{cases} \tag{3.4}$$

Figure 3.3 Computation of Frequency Counts on AVT

parentheses). Finally, we update the estimated frequency counts for all attribute values as shown in Figure 3.4-(D).

3.2.2 Construction of AVT-guided Decision Tree

As we have described earlier, standard decision tree algorithm considers partitioning the data based on the values of each candidate attribute, selecting the most informative attribute at each step. However, AVT-DTL has to choose not just the attribute, but also, the appropriate level in the taxonomy which defines the values of the attribute on which to partition the data. This is done by keeping track of "pointing vectors" in the AVTs during the process of decision tree construction.

For each node in the decision tree, we maintain a pointing vector, a set of pointers that point to the nodes (attribute values) in the corresponding AVTs, which also correspond to a

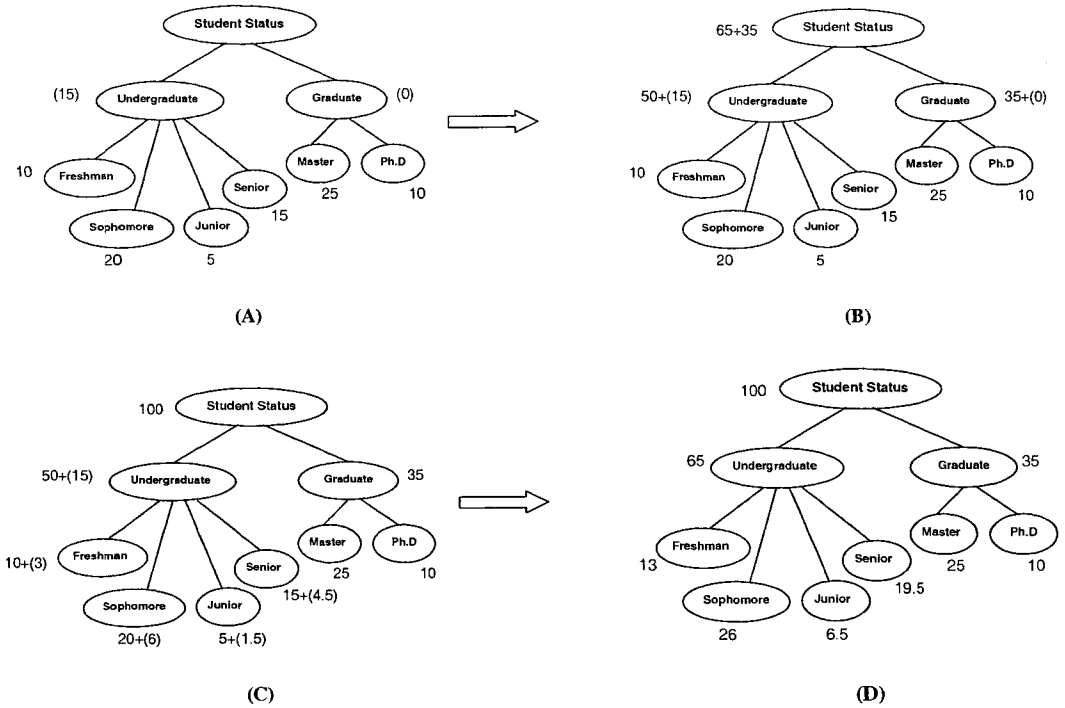


Figure 3.4 Estimation of class conditional frequency counts. (A) Initial counts associated with each attribute value showing the number of positively labeled instances. (B) Aggregation of counts upwards from each node to its ancestors. (C) Distribution of counts of a partially specified attribute value downwards among descendant nodes. (D) Updating the estimated frequency counts for all attribute values.

conceptual frontier in AVTs. During the learning phase, we are pushing the frontier defined by the pointing vectors minimally to achieve accurate classification of training data. Therefore, at each step of the tree construction, we are seeking tests of abstract values to make sufficient informative test for the current set of instances, which, as in the case of standard decision tree, yields the maximum reduction in the impurity measurement. The standard decision tree learning algorithm can only have pointing vectors at one level (which would point to the attribute values without specified AVTs), while our AVT-DTL algorithm is able to keep track of pointing vectors at different levels of abstraction in AVTs to decide the optimal concept frontier for the classifier.

As an example, in Figure 3.5, the pointing vector points to two high-level attribute values in the two corresponding taxonomies.

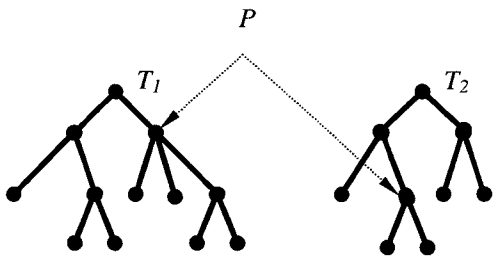


Figure 3.5 Illustration of a Pointing Vector P

More precisely, let $P_S = \{p_1^{(S)}, p_2^{(S)}, \cdots, p_N^{(S)}\}$ be a pointing vector for a set of instances S where $p_i^{(S)}$ is a pointer to a value in $\mathcal{T}_i$ of attribute A_i. We choose the best pointer $p_a^{(S)} \in P_S$ to partition S that yields the maximum reduction in impurity measurement (e.g., information gain, gain ratio, etc.), and uses attribute values in $\pi(p_a^{(S)}, \mathcal{T}_i)$ to generate partition subsets $\{S_i | i = 1, \cdots, |\pi(p_a^{(S)}, \mathcal{T}_i)|\}$. Each subset S_i has to update a new pointing vector P_{S_i} by replacing $p_a^{(S)}$ with a corresponding child attribute value in $\pi(p_a^{(S)}, \mathcal{T}_i)$. This process is recursively iterated until either $\Phi(P_S) = true$ or all instances in S have the same class label (S is a pure set).

One noticeable feature of AVT-DTL is that the union of pointing vectors of all leaves in an intermediate decision tree corresponds to a valid cut Γ in AVTs. AVT-DTL indirectly make refinement on Γ at each stage of decision tree construction. This Γ has been pushed minimally to achieve accurate classification of data. Also notice that when pointing vectors come down to the leaf nodes of the decision tree, they directly represent the classification rules in a conjunction form.

The main procedure of AVT-DTL can be summarized in Figure 3.6.

The AVT-DTL algorithm performs a AVT-guided hill-climbing search in a decision tree hypothesis space. Because AVT-DTL makes explicit the attribute value taxonomies associated with attributes, it enables users to explore data from multiple perspectives and multiple levels of abstraction.

Constructing AVT-guided Decision Tree:

Tree-Builder(S, P_S)

1. *If* each instance in S has same label C_i, return(C_i); *else* if $\Phi(P_S)$ is true then assign majority label to generate a leaf node in decision tree (Stopping Criterion).

2. For each pointer $p_i^{(S)}$ in P_S do:

   a) Check each partially specified instance with partially specified value v for attribute A_i with the pointer $p_i^{(S)}$. If $depth(\mathcal{T}_i, p_i^{(S)}) < depth(\mathcal{T}_i, v)$, then the partially specified instance is treated as though it were a fully specified instance and sent along the appropriate branch of the decision tree rooted at $p_i^{(S)}$. Otherwise (that is, if $depth(\mathcal{T}_i, p_i^{(S)}) \geq depth(\mathcal{T}_i, v)$), break v into $|\pi(v, \mathcal{T}_i)|$ fractional instances according to the distribution of counts associated with the elements in $\pi(v, \mathcal{T}_i)$. Each factional instance v_f, where f is an element of $\pi(v, \mathcal{T}_i)$, is assigned with a weight that is proportional to the following probability:

$$Weight(v_f) = Pr(f) = \frac{\sigma_i(f, \mathcal{T}_i)}{\sum_{e \in \pi(v, \mathcal{T}_i)} \sigma_i(e, \mathcal{T}_i)} \tag{3.5}$$

   b) Calculate the entropy of the set S based on the partition by $\pi(p_i^{(S)}, \mathcal{T}_i)$ in S.

3. Choose the best pointer $p_a^{(S)}$ in P_S to partition S that yields the maximum information gain.

4. Partition the sample set S into subsets $S_1, S_2, \cdots, S_{|\pi(p_a^{(S)}, \mathcal{T}_i)|}$ by using attribute values in $\pi(p_a^{(S)}, \mathcal{T}_i)$.

5. Extend P_S to obtain a new pointing vector corresponding to each of the subsets $S_1, S_2, \cdots, S_{|\pi(p_a^{(S)}, \mathcal{T}_i)|}$ by replacing the pointer $p_a^{(S)}$ with the value of attribute A_i in the corresponding subset S_j, where $1 \leq j \leq \pi(p_a^{(S)}, \mathcal{T}_i)$.

6. For each $j \in \{1, \cdots, |\pi(p_a^{(S)}, \mathcal{T}_i)|\}$, Do *Tree-Builder*($S_j$, P_{S_j}).

Figure 3.6 Construction of AVT-guided Decision Tree

Because AVT-DTL has a preference to choose split on more abstract values at higher levels of AVTs, the decision tree that has been built by AVT-DTL is generally more compact than that built by standard decision tree learning algorithms. This smaller tree corresponds to less rules, which could be the generalizations of several specific rules produced by standard decision learning algorithm. Moreover, compact decision trees with less rules are easily comprehensible to humans.

As a demonstrative example, we use the same customer purchase data as shown in Figure 3.1, and for each attribute we define a corresponding attribute value taxonomy. Figure 3.7 shows the two AVTs for Beverage (Item 1) and Snack (Item 2). For concepts in the Beverage taxonomy, there are three different levels of abstraction, and in the Snack taxonomy, we have two different levels of abstraction. Figure 3.8 shows the induced decision tree.

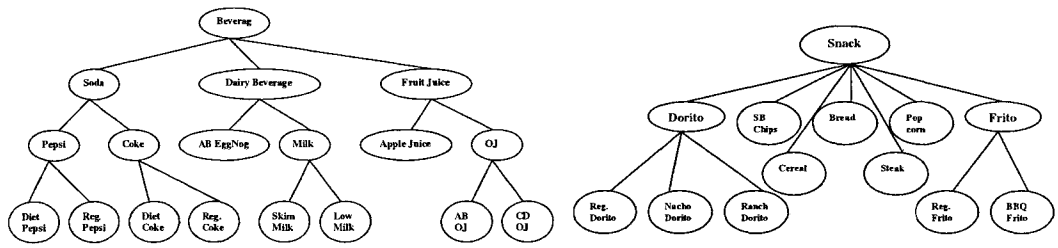


Figure 3.7 Two AVTs defined over the attributes of Beverage and Snack

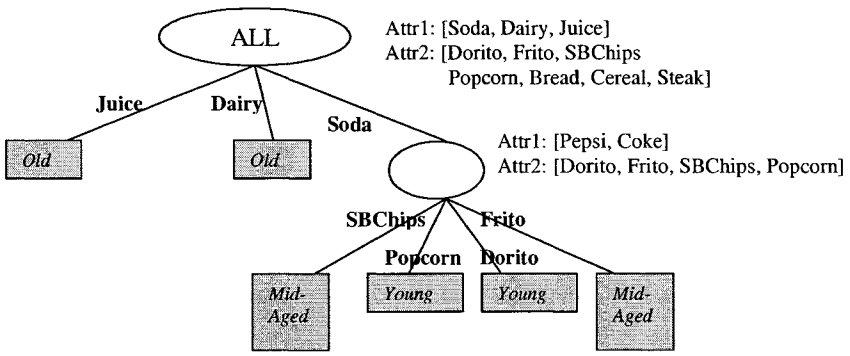


Figure 3.8 Decision Tree built on the customer purchase data set by AVT-DTL

We start our search with the pointing vectors pointing to the root of both taxonomies.

The information gain associated with the attribute Beverage is higher than that associated with Snack. Consequently, the attribute corresponding to the root of the Beverage hierarchy is selected to partition the original data. This yields a 3-way split at the root of the decision tree. Two examples are classified as Old on the basis of the attribute corresponding to the concept "Dairy Beverage", and two examples are classified as Old on the basis of the attributed corresponding to the concept "Fruit Juice". The remaining eight examples need to be partitioned further. The first element p_1 in Pointing Vector for these eight examples changes to "Soda" in the Beverage Taxonomy, and therefore the possible choice of attribute values will be [Pepsi, Coke]. While the second element p_2 continues to the root of the Snack Taxonomy, the possible attribute value set will include all the available concepts in the first level of this taxonomy. This time p_2 yields a better value for information gain and all eight examples are correctly classified. The resulting decision tree corresponds to the following high level rules: *If* Soda and Dorito *Then* Young; *If* Soda and Frito *Then* Middle-aged; *If* Dairy *Then* Old; *If* Juice *Then* Old; etc. We obtained a compact decision tree with less rules (8 nodes and 6 rules) comparing to the decision tree (13 nodes and 10 rules) shown in Figure 3.2), which is constructed by standard decision tree learning algorithm. Note that the two trees have the same 100% accuracy on the training data. The high level rules (i.e., rules using concepts that reside at higher levels of the corresponding attribute value taxonomies) represented in the compact tree are more comprehensible than lower level rules (i.e., rules using primitive attribute values only), and high level rules actually summarize several lower level rules. For example, the rule "*If* Soda and Frito *Then* Middle-aged" is a summarization of two separate rules: "*If* RegPepsi and RegFrito *Then* Middle-aged", and "*If* RegPepsi and BBQFrito *Then* Middle-aged". This is true because "Frito" is an abstraction of both "RegFrito" and "BBQFrito", and "Soda" is an abstraction of "RegPepsi".

3.2.3 Handling Partially Specified Data in AVT-DTL

The AVT-DTL algorithm has an embedded mechanism to deal with partially specified data. It generalizes existing approaches to dealing with missing attribute values and applies

to partially specified values in both decision tree construction and classification. Based on the AVT frontier being explored by pointing vectors, it is necessary to differentiate two cases in handling partially specified value:

I. The value in an instance is a descendent of the node in the AVT that is pointed to by the current pointer (as shown in Figure 3.9(a)). This partially specified instance is treated as a fully specified instance because the candidate split uses a pointer above the level of this abstract value.

II. The value in an instance is the same node in the AVT that is pointed to by the current pointer of candidate split (as shown in Figure 3.9(b)). We treat the value as partially missing, and we have two options in handling this partially specified value in our implementation of AVT-DTL.

   - One option is to replace this value probabilistically with one descendant value according to statistical distribution among the descendants of the current pointer, and use the filled in value to compute the contribution of corresponding instance to entropy calculation for the split being considered. This approach has been implemented in an early version of AVT-DTL.
   - Another option is to break the instance with partially specified value into fractional instances with its descendant values. Each fractional instance is assigned a weight that corresponds to its estimated probability in the distribution among the descendant values. All fractional instances will be distributed into different partition sets based on the current split, and then compute the contributions made by corresponding fractional instances to entropy calculation. This approach has been implemented in the current version of AVT-DTL.

Therefore, any attribute value can dynamically switch between a fully specified value and a partially specified value with regard to the current AVT frontier (or a global cut) that is being explored by the learning algorithm. It all depends on the information that a particular instance can contribute to the learner. If it provides the detailed information about the attribute value

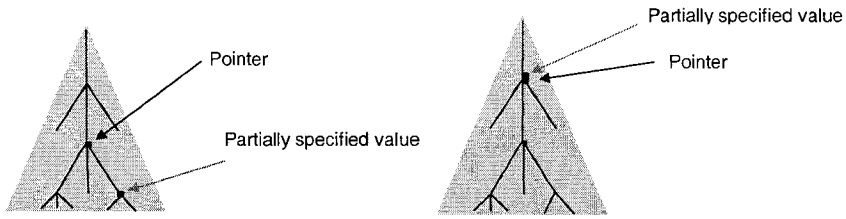


Figure 3.9 Two different cases when handling partially specified data

for building the classifier, it will be treated as fully specified. However, if the attribute value is above the AVT frontier that is being explored, then we lack of the information needed by the learner, and we have to use probabilistic distribution estimations from the original data to estimate the contribution of this partially specified value to the construction of the classifier. A specific attribute value can be fully specified initially, but when learning algorithms start to making refinement on the AVT frontier, and the level of abstraction can be below the attribute value, it becomes partially specified, because no further information can be provided about this value with regard to the AVT frontier. As we have already shown, AVT-DTL is equipped with the ability to estimate the probability distribution of attribute values in AVTs when data are partially specified, and is able to handle partially specified data during both the learning phase and classification phase of the decision tree classifier.

3.3 Alternative Approaches and Further Analysis

There is an alternative to the proposed AVT-based learning algorithm (e.g., AVT-DTL). We can consider applying standard learning algorithms to suitably preprocessed data sets by augmenting or transforming the original set of attributes using information provided by the AVTs. We will consider approaches that can transform a partially specified instance into a set of fully specified instances so that we can apply standard learning algorithms directly to partially specified data as well. These preprocessing methods normally either loose information on partially specified data or increase the size of the original data set and the number of available features, which makes learning less efficient. In contrast, our AVT-guide learning approaches are more efficient and free of such a computational expensive load. We will analyze

the computational complexity and compare the performance of both approaches. We will consider the following two alternative approaches in learning classifiers from attribute value taxonomies and partially specified data:

Approaches that Ignore Attribute Value Taxonomy. This approach reduces to the standard learning algorithm and it lacks the ability to deal with partially specified data. All partially specified instance will be treated as completely missing, and the resulting data set with missing values is handled using standard approaches for dealing with missing attribute values in learning decision tree classifier. Because of ignoring AVT as the choice of working assumptions, it fails to capture the relevant relations among attribute values in the generation of simple and accurate classifiers from data. Though often associated with poor performance by ignoring AVT, a main advantage of this approach is that it requires no modification to the learning algorithm. All we need is a simple preprocessing step in which all partially specified attribute values are turned into missing attribute values.

AVT-Based Propositionalization Methods. The data set is represented using a set of Boolean attributes obtained from $Nodes(\mathcal{T}_i)$ of attribute A_i by associating a Boolean attribute with each node (except the root) in $\mathcal{T}_i$. Thus, each instance in the original data set defined using N attributes is turned into a Boolean instance specified using $\tilde{N}$ Boolean attributes where $\tilde{N} = \sum_{i=1}^{N} |Nodes(\mathcal{T}_i)|$. The Boolean attributes that correspond to descendants of the partially specified attribute value are treated as unknown.

In the case of the *student status* taxonomy shown in Figure 2.3, this would result in binary features that correspond to the propositions such as (student = *Undergraduate*), (student=*Graduate*), (student=*Freshman*), ... (student=*Senior*), (student=*Master*), (student=*Ph.D*). Based on the specified value of an attribute in an instance e.g., (student = *Master*), the values of its ancestors in the AVT (e.g., student=*Graduate*) are set to True because the AVT asserts that *Master* students are also *Graduate* students. But the Boolean attributes that correspond to descendants of the specified attribute value are treated as unknown. For example, when the value of the *student status* attribute is partially specified in an instance, e.g. (student=*Graduate*), the corresponding Boolean attribute is set to True, but the Boolean at-

tributes that correspond to the descendants of *Graduate* in the this taxonomy are treated as *missing*. The resulting data with some missing attribute values can be handled using standard approaches to dealing with missing attribute values. For numerical attributes, the Boolean attributes are the intervals that correspond to nodes of the respective AVTs. If a numerical value falls within a certain interval, the corresponding Boolean attribute is set to True, otherwise it is set to False. We call the resulting algorithm - DTL applied to AVT-based propositionalized version of the data - Prop-DTL.

Note that the Boolean features created by the propositionalization technique described above are not independent given the class. A Boolean attribute that corresponds to any node in an AVT is necessarily correlated with Boolean attributes that correspond to its descendants as well as its ancestors in the tree. For example, the Boolean attribute (student=*Graduate*) is correlated with (student=*Master*). (Indeed, it is this correlation that enables us to exploit the information provided by AVT in learning from partially specified data). This could degrade the performance of those classifiers (e.g., Naïve Bayes classifier) that rely heavily on the independence of attributes given class. The statistical dependence among the Boolean attributes in the propositionalized representation of the original data set can also degrade the performance of other classifiers.

A main advantage of the AVT-based propositionalization methods is that they require no modification to the learning algorithm. However it does require preprocessing of partially specified data using the information supplied by an AVT. The number of attributes in the transformed data set is substantially larger than the number of attributes in the original data set.

We now analyze the time complexity of AVT-DTL, and compare it with that of propositionalization approach. Within a particular attribute A_i, We denote K the number of nodes in the AVT, and denote L the number of leafs in the AVT (primitive attribute values). Let $|D|$ be the number of training instances, and M be the number of classes.

Let us consider applying Prop-DTL first. Computing the class frequencies for each Boolean attribute takes time proportional to $|D|$, and computing information gain takes time propor-

tional to M. Because the total number of Boolean attributes within this attribute is K, the time complexity of processing this attribute is $O(K \cdot (|D| + M))$, which can be written as $O(K \cdot |D|))$ because of $|D| \gg M$.

According to the description of AVT-DTL, at each step, we calculate the entropy based on the partition by $\pi(v, \mathcal{T}_i)$, the children elements of a pointer v. The size of $\pi(v, \mathcal{T}_i)$ is dramatically smaller than the number K, and is also smaller than L, that is, $|\pi(v, \mathcal{T}_i)| < L \ll K$. The time complexity of processing this attribute by AVT-DTL is $O(|\pi(v, \mathcal{T}_i)| \cdot (|D| + M))$, and can be simplified to $O(|\pi(v, \mathcal{T}_i)| \cdot |D|)$. Thus, AVT-DTL is much more efficient than Prop-DTL, and is also time efficient comparing to standard DTL.

3.4 Performance Study

In order to systematically assess our proposed learning algorithm, we need to make empirical evaluations on the resulting classifiers. Specifically, we need to test the accuracy and generalization ability of the classifiers, the complexity of the classifiers, and the robustness in the presence of partially specified or totally missing attribute values.

Although data sets with partially specified attribute values and AVT are commonly encountered in practice, benchmark data sets with predefined AVTs and partially specified attribute value are not readily available in collections of benchmark data sets. This could be the major reason for the lack of systematic study and evaluation of AVT-based learning algorithms.

Therefore, we need to prepare data sets to facilitate controlled exploration of the performance of algorithms for learning from AVT and partially specified data.

3.4.1 Data Preparation

UCI Data Repository (http://www.ics.uci.edu/~ mlearn/MLRepository.html) offers a widely used collection of benchmark data sets for research in machine learning and knowledge discovery. Most data sets in the UCI repository do not have associated AVTs. Hence, in experimenting with UCI datasets, we need to specify a reasonable set of AVTs by manually or automatically grouping related attribute values to generate hierarchical taxonomies of values.

We also need to generate data sets with different percentages of totally missing or partially missing attribute values in order to test the performance of our algorithm in dealing with partially specified data.

We selected a collection of data sets from the UC Irvine Machine Learning Repository. For only three of the data sets (i.e., *Mushroom Toxicology*, *Soybean*, and *Nursery*), AVTs were available. In the case of *Mushroom Toxicology*, the AVT was supplied by a botanist. For *Soybean* and *Nursery* data, the AVTs were specified based on our understanding of the domain. For the remaining data sets, no expert-generated AVTs are readily available. Hence, the AVTs on both nominal and numerical attributes were generated using AVT-Learner [Kang et al. (2004)], a Hierarchical Agglomerative Clustering algorithm to construct AVTs for learning. Then, data sets with pre-specified percentage of totally or partially missing attribute values were generated by assuming that the missing values are uniformly distributed on the attributes based on the constructed AVTs.

3.4.1.1 Learning AVTs from Data

Next, we briefly describe the AVT-Learner [Kang et al. (2004)], an algorithm for automated construction of AVT from a data set.

For each attribute A_i, we generate $\mathcal{T}_i$ by a hierarchical grouping of values in A_i according to the specified similarity measure. This hierarchical grouping process is based on the principle of Hierarchical Agglomerative Clustering (HAC), which groups attribute values according to the distribution of classes that co-occur with them. Let $DM\left(P\left(x\right)||P\left(y\right)\right)$ denote a measure of pairwise divergence between two probability distributions $P\left(x\right)$ and $P\left(y\right)$ where the random variables x and y take values from the same domain. We use the pairwise divergence between the distributions of class labels associated with the corresponding attribute values as a measure of the dissimilarity between the attribute values. The lower the divergence between the class distributions associated with two attributes, the greater is their similarity.

The basic idea behind AVT-Learner is to construct an AVT $\mathcal{T}_i$ for each attribute A_i by starting with the primitive values in V_i as the leaves of $\mathcal{T}_i$ and recursively add nodes to $\mathcal{T}_i$

one at a time by merging two existing nodes. To aid this process, the algorithm maintains a cut γ_i through the AVT $\mathcal{T}_i$, updating the cut γ_i as new nodes are added to $\mathcal{T}_i$. At each step, the two attribute values to be grouped together to obtain an abstract attribute value to be added to $\mathcal{T}_i$ are selected from γ_i based on the divergence between the class distributions associated with the corresponding values. That is, a pair of attribute values in γ_i are merged if they have more similar class distributions than any other pair of attribute values in γ_i. This process terminates when the cut γ_i contains a single value which corresponds to the root of $\mathcal{T}_i$. Among various divergence measures, we choose Jensen-Shannon divergence measure [Slonim and Tishby (1999)]. The Jensen-Shannon divergence is the weighted information gain, and it is reflexive, symmetric and bounded. It is given by:

$$I(P||Q) = \frac{1}{2}\left[\sum p_i log\left(\frac{2p_i}{p_i+q_i}\right) + \sum q_i log\left(\frac{2q_i}{p_i+q_i}\right)\right]$$

In the case of continuous-valued attributes, we define intervals based on observed values for the attribute in the data set. We then generate a hierarchical grouping of adjacent intervals, selecting at each step two adjacent intervals to merge using the pairwise divergence measure. A cut through the resulting AVT corresponds to a discretization of the continuous-valued attribute.

The pseudo-code of AVT-Learner is shown in Figure 3.10. Note that v_i^j is an attribute value in A_i, and γ_i represents a cut through AVT $\mathcal{T}_i$.

3.4.1.2 Generating Data Sets with Partially/Totally Missing Attribute Values

In order to explore the performance of AVT-DTL on data sets with different percentages of totally missing or partially missing attribute values, data sets with a pre-specified percentage (e.g., 10%, 30%, or 50%, excluding the missing values in the original data set) of (totally) missing attribute values were generated by assuming that the missing values are uniformly distributed on the attributes as well as on the instances. Data sets were generated for each choice of percentage of missing values. From the original data set D, a data set D_p of partially missing values was generated as follows: Let $(n_l, n_{l-1}, \cdots, n_0)$ be the path from the fully

AVT-Learner:
begin
Input : data set D
For each attribute A_i:
  For each attribute value v_i^j :
    For each class label c_k: estimate the probability $p\left(c_k|v_i^j\right)$
    Let $P\left(C|v_i^j\right) = \left\{p\left(c_1|v_i^j\right), \ldots, p\left(c_k|v_i^j\right)\right\}$ be the class distribution given the value .
    Set $\gamma_i \leftarrow V_i$; Initialize $\mathcal{T}_i$ with nodes in γ_i.
  Iterate until $|\gamma_i| = 1$:
    In γ_i, find $(x, y) = argmin\left\{DM\left(P\left(C|v_i^x\right)||P\left(C|v_i^y\right)\right)\right\}$
    Merge v_i^x and v_i^y $(x \neq y)$ to create a new value v_i^{xy}.
    Calculate probability distribution $P\left(C|v_i^{xy}\right)$.
    $\gamma_i \leftarrow \gamma_i \cup \{v_i^{xy}\} \setminus \{v_i^x, v_i^y\}$.
    Update $\mathcal{T}_i$ by adding nodes v_i^{xy} as a parent of v_i^x and v_i^y.
Output : $T = \{\mathcal{T}_1, \mathcal{T}_2, \cdots, \mathcal{T}_n\}$
end.

Figure 3.10 Pseudo-code of AVT-Learner

specified primitive value n_l to the root n_0 of the corresponding AVT; Select one of the nodes (excluding n_l) along this path with uniform probability; Read the corresponding attribute value from the AVT and assign it as the partially specified value of the corresponding attribute. Note that the selection of the root of the AVT would result in a totally missing attribute value, which can be recorded as "?". Similarly, we generate a data set D_t of totally missing values with a pre-specified percentage from D_p by replacing all partially specified attribute values with "?". Therefore, corresponding to every instance $I \in D_p$ that has a partially specified value for some attribute (say the jth attribute), there is a corresponding instance in D_t in which the jth attribute is totally missing.

3.4.2 Experiments

The goal of our performance study on AVT-DTL is to answer the following two questions:

(1) How does AVT-DTL compare with standard Decision Tree Learning algorithm (e.g., C4.5) and its variants with respect to compactness and comprehensibility of the resulting classifiers?

(2) How does AVT-DTL compare with DTL with increasing percentages of partially missing

attribute values? Can AVT-DTL efficiently deal with partially specified data?

To carry out performance evaluations, we designed the following sets of experiments:

The first set of experiments compares the performance of AVT-DTL with standard DTL (for standard decision tree learning algorithm, we choose C4.5 [Quinlan (1993)]), and Prop-C4.5 (i.e., C4.5 applied to propositionalized data sets). We also use C4.5 with 'subsetting' option (i.e., '-s' option) to compare with AVT-DTL. The 'subsetting' option allows C4.5 to consider splits based on subsets of attribute values (as opposed to single values) along each branch. We provide further discussions on C4.5 with 'subsetting' in related work at the end of this chapter. For each approach, we construct decision trees with and without pruning. Whenever pruning was applied, we chose reduced error pruning, which assesses the error rates of the tree and its components directly on the set of separate validation samples [Quinlan (1987)].

The second set of experiments is to test the performance of AVT-DTL on data sets with different percentages of totally missing and partially missing attribute values. Data sets with a pre-specified percentage (excluding the missing values in the original data set) of totally or partially missing attribute values were generated.

The third set of experiments is to study the learning curve of AVT-DTL, and to explore the efficiency and robustness of AVT-DTL. We compare the performance of classifiers generated by AVT-DTL and C4.5 as a function of the training set size. We divided each data set into two disjoint parts: a training pool and a test pool. Training sets of different sizes were sampled and used to train decision tree classifier using AVT-DTL and C4.5. The resulting classifiers were evaluated on the entire test pool.

For each set of experiments, the error rate of the resulting decision tree was normally estimated using 10-fold cross-validation, and we calculate 90% confidence interval on the error rate. The reported size of the decision tree corresponds to the average size computed from the 10 experiments.

Table 3.2 Comparison of tree size and number of leaves in decision trees built by variants of C4.5, Prop-C4.5 and AVT-DTL in original data set. We use the following abbreviations: C4.5 - standard decision tree learning algorithm without pruning; C4.5P - C4.5 with pruning; C4.5S - C4.5 with subsetting; C4.5SP - C4.5 with subsetting and pruning; Prop-C4.5 - C4.5 applied to propositionalized data; Prop-C4.5P - C4.5 applied to propositionalized data with pruning; AVT-DTL - AVT-DTL without pruning; AVT-DTLP - AVT-DTL with pruning. Whenever pruning is applied, we use reduced error pruning.

	Mushroom		Nursery	
	Tree Size	Number of Leaves	Tree Size	Number of Leaves
C4.5	31	26	944	680
C4.5P	31	26	511	359
C4.5S	20	12	455	272
C4.5SP	15	9	327	168
Prop-C4.5	21	11	391	196
Prop-C4.5P	21	11	333	131
AVT-DTL	16	10	298	172
AVT-DTLP	16	10	223	122

3.4.3 Results

Our experimental results can be summarized as follows.

3.4.3.1 AVT-DTL produces compact and easy-to-comprehend decision tree classifiers

Table 3.2 summarizes the results of the experiments comparing AVT-DTL with several variants of C4.5 on the original *Mushroom Toxicology* and *Nursery* data (without any partially missing values). AVT-DTL, even without pruning, yields substantially smaller trees compared to standard C4.5 (with or without pruning) and C4.5 with 'subsetting' but no pruning. Furthermore, the smaller tree size is achieved by AVT-DTL without any deterioration in predictive accuracy. For *Mushroom Toxicology* data, C4.5 (with or without pruning) yields a tree with 31 nodes which correspond to 26 rules, each of which includes tests on attribute values cor-

responding to the lowest levels of the AVTs. In contrast, AVT-DTL produces a decision tree with 16 nodes which corresponds to 10 rules in which tests correspond to attribute values at the higher levels of the AVTs. C4.5 with 'subsetting', but without pruning, produces trees that are substantially larger than those obtained by AVT-DTL. It is only when C4.5 uses both subsetting and pruning that the resulting trees are comparable in size (as measured by the number of nodes or the number of leaves) or smaller than those obtained by AVT-DTL. This can be explained by the fact that C4.5 with 'subsetting' is less constrained in the choice of splits at each node compared to AVT-DTL (whose splits are constrained by the AVT). However, a decision tree built by C4.5 with 'subsetting' is hard to comprehend from the point of view of the users. Prop-C4.5 generates smaller trees compared to C4.5 with 'subsetting', but generates slightly larger trees compared to C4.5 with 'subsetting' and pruning. We can apply pruning to Prop-C4.5 as well, and expect more compact trees.

A representative decision tree generated by AVT-DTL on *Mushroom Toxicology* data is shown in the Figure 3.11. AVT-DTL is quite effective in selecting tests based on attribute values from the higher levels of AVT (and hence correspond to more abstract attribute values) e.g., **if** (*Odor* = *none*) AND (*Spore_print_color* = *dark*) then *edible*, and *dark* is an abstract attribute value corresponding to a set of colors: *black, brown, chocolate*. Examination of decision trees generated using C4.5 shows that this rule in fact summarizes three separate rules generated using C4.5: **if** (*Odor* = *none*) AND (*Spore_print_color* = *black*) then *edible*; **if** (*Odor* = *none*) AND (*Spore_print_color* = *brown*) then *edible*; and **if** (*Odor* = *none*) AND (*Spore_print_color* = *chocolate*) then *edible*.

3.4.3.2 AVT-DTL yields more accurate decision tree classifiers

Table 3.3 shows the error rate estimates of the decision tree classifiers generated by C4.5, Prop-C4.5 and AVT-DTL on UCI benchmark data sets without applying pruning. Table 3.4 shows the results on the same data sets with reduced error pruning. As indicated by the results, the error rate of AVT-DTL is substantially smaller than that of C4.5, and is also obviously smaller than that of Prop-C4.5. Prop-C4.5 sometimes can produce even more compact trees

Table 3.3 Comparison of error rate and size of decision tree classifiers generated by C4.5, PROP-C4.5, and AVT- DTL on several benchmark data sets. The error rates and the sizes were estimated using 10-fold cross validation with 90% confidence interval. We use the number of leaves in the tree to represent the size of the decision tree classifier. No pruning is applied.

DATA SET	C4.5		PROP- C4.5		AVT- DTL	
	ERROR	*SIZE*	ERROR	*SIZE*	ERROR	*SIZE*
Breast-Cancer	33.91(±5.06)	*152*	32.86(±5.03)	*58*	29.37(±4.87)	*38*
Car	7.75(±1.16)	*297*	1.79(±0.58)	*78*	1.67(±0.57)	*78*
Dermatology	6.83(±2.38)	*71*	5.74(±2.20)	*19*	5.73(±2.19)	*22*
Mushroom	0.0(±0.00)	*26*	0.0(±0.00)	*10*	0.0(±0.00)	*10*
Nursery	3.34(±0.90)	*680*	1.75(±0.66)	*196*	1.21(±0.55)	*172*
Soybean	9.81(±2.06)	*175*	8.20(±1.90)	*67*	7.75(±1.85)	*90*
Zoo	7.92(±4.86)	*13*	8.91(±5.13)	*9*	7.92(±4.86)	*7*

Table 3.4 Comparison of error rate and size of decision tree classifiers generated by C4.5, PROP-C4.5, and AVT- DTL on several benchmark data sets with pruning. The error rates and the sizes were estimated using 10-fold cross validation with 90% confidence interval. We use the number of leaves in the tree to represent the size of the decision tree classifier. Reduced error pruning is applied.

DATA SET	C4.5		PROP- C4.5		AVT- DTL	
	ERROR	*SIZE*	ERROR	*SIZE*	ERROR	*SIZE*
Breast-Cancer	35.62(±5.15)	*38*	34.26(±5.07)	*30*	29.37(±4.87)	*4*
Car	10.82(±1.35)	*80*	3.94(±0.85)	*39*	3.76(±0.83)	*40*
Dermatology	6.28(±2.29)	*22*	4.15(±1.89)	*8*	4.13(±1.88)	*12*
Mushroom	0.0(±0.00)	*26*	0.0(±0.00)	*10*	0.0(±0.00)	*10*
Nursery	5.51(±1.14)	*680*	3.16(±0.88)	*131*	2.89(±0.84)	*122*
Soybean	12.01(±2.25)	*85*	10.83(±2.15)	*35*	11.42(±2.20)	*39*
Zoo	7.92(±4.86)	*12*	7.92(±4.86)	*8*	7.92(±4.86)	*7*

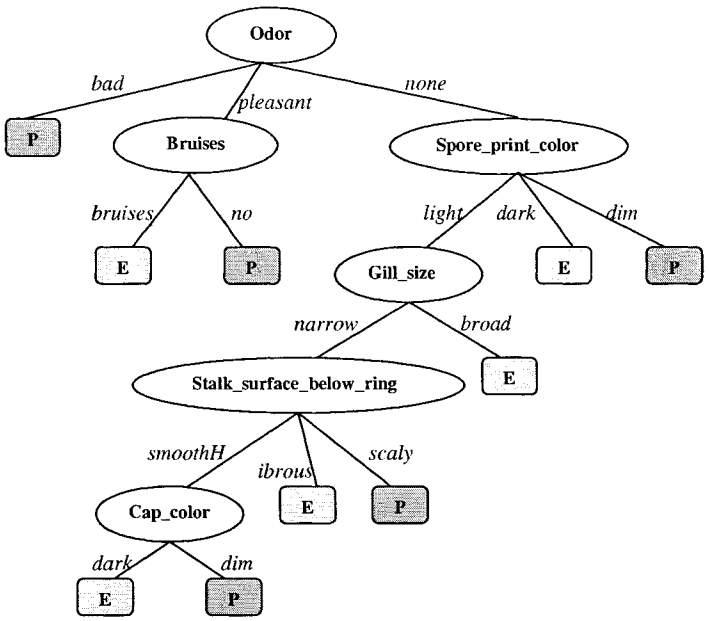


Figure 3.11 A decision tree learned by AVT-DTL from the *Mushroom Toxicology* data

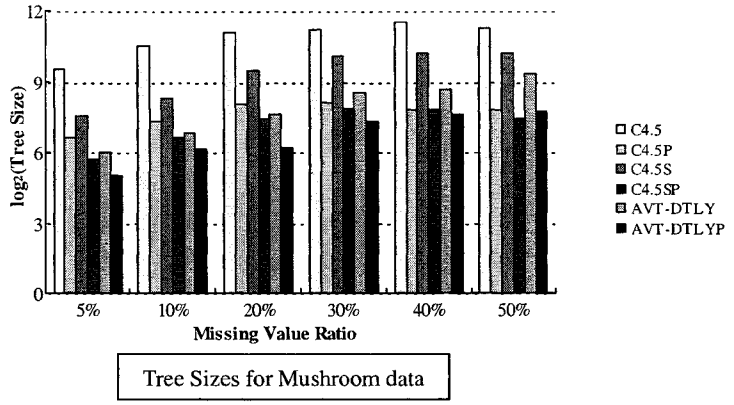
than does AVT-DTL. However, this has been done in the cost of accuracy. Overall, AVT-DTL achieves a better trade-off compared to Prop-C4.5.

As shown in the table, reduced error pruning does reduce the sizes of the trees, however, it often leads to less accurate trees as well. The main drawback associated with reduced error pruning is that a separate validation set needs to be reserved for pruning, which reduces the amount of data that is available to construct the tree. When data are limited, this could deteriorate the accuracy of the classifiers. Although cross-validation is performed here, it still increases the number of observed errors compared to its non-pruning counterpart.

3.4.3.3 AVT-DTL yields significantly lower error rates than C4.5 on data sets with substantially large percentage of partially missing attribute values

Table 3.5 shows the resulting error estimates along with the corresponding 90% confidence intervals. Note that all algorithms, except for AVT-DTL, treat a partially missing attribute value as a totally missing attribute value during the decision tree construction because they

do not utilize AVTs.



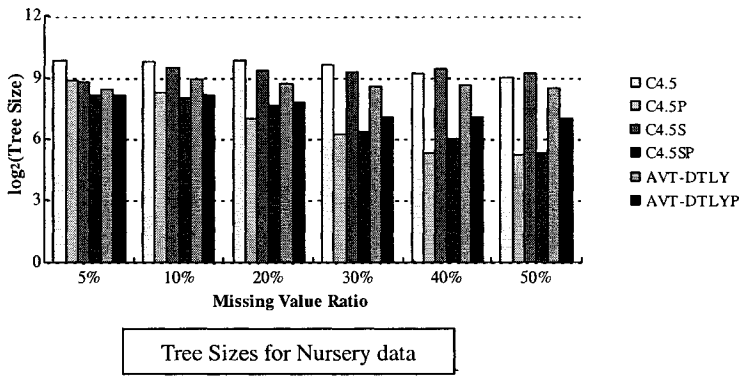


Figure 3.12 Comparison of the sizes of the induced decision trees with different percentages of partially missing values. Note that the Y axis shows the logarithm of the tree size to base 2. We use the same abbreviations as previously defined.

In the case of Mushroom Toxicology data, the error rate of AVT-DTL (5.58% without pruning and 6.51% with pruning) is substantially smaller than error rates of each of the variants of C4.5 (which range from 15.92% in the case of C4.5 with subsetting, but no pruning, to 24.04% in the case of C4.5 with pruning) when half (50%) of the attribute values (excluding the attribute values that were originally totally missing in the data set) are partially missing. Qualitatively similar results are obtained in the case of Nursery data as well. This is consistent with the fact that AVT-DTL constructs decision trees that favor tests on abstract attributes

Table 3.5 The error rate estimates for AVT-DTL and C4.5 (with different options) with different percentages of partially and totally missing values. We use the following abbreviations: C4.5 - standard decision tree learning algorithm without pruning; C4.5P - C4.5 with pruning; C4.5S - C4.5 with subsetting; C4.5SP - C4.5 with subsetting and pruning; AVT-DTLT - AVT-DTL without applied to data sets with totally missing values; AVT-DTLTP - AVT-DTL with pruning applied to data sets with totally missing values. AVT-DTLY - AVT-DTL without pruning applied to data sets with partially missing values; AVT-DTLYP - AVT-DTL with pruning applied to data sets with partially missing values. The error rates were estimated using 10-fold cross validation, and we calculate 90% confidence interval on each error rate.

Percentage of totally or partially missing values		0%	5%	10%	20%	30%	40%	50%
Mushroom toxicology Data	C4.5	0.0±(0.00)	0.99±(0.64)	2.01±(0.90)	4.22±(1.27)	8.08±(1.73)	14.21±(2.21)	22.95±(2.69)
	C4.5P	0.0±(0.00)	1.03±(0.62)	2.16±(0.91)	5.31±(1.42)	12.46±(2.09)	16.26±(2.33)	24.04±(2.70)
	C4.5S	0.0±(0.00)	0.99±(0.64)	1.68±(0.81)	2.87±(1.06)	7.14±(1.63)	10.75±(1.96)	15.92±(2.32)
	C4.5SP	0.0±(0.00)	0.99±(0.64)	1.90±(0.86)	3.70±(1.19)	8.90±(1.80)	12.60±(2.11)	18.80±(2.48)
	AVT-DTLT	0.0±(0.00)	0.94±(0.60)	1.72±(0.82)	2.99±(1.08)	8.73±(1.79)	12.08±(2.07)	20.36±(2.55)
	AVT-DTLTP	0.0±(0.00)	0.95±(0.61)	1.97±(0.87)	3.84±(1.21)	9.78±(1.88)	13.62±(2.17)	22.45±(2.64)
	AVT-DTLY	0.0±(0.00)	0.47±(0.43)	1.42±(0.75)	2.18±(0.91)	3.19±(1.11)	4.09±(1.24)	5.58±(1.45)
	AVT-DTLYP	0.0±(0.00)	0.52±(0.45)	1.72±(0.82)	2.59±(1.00)	3.94±(1.23)	4.87±(1.36)	6.51±(1.56)
Nursery Data	C4.5	3.34±(0.90)	10.03±(1.51)	14.77±(1.78)	22.11±(2.08)	30.01±(2.30)	36.32±(2.41)	41.79±(2.47)
	C4.5P	5.51±(1.14)	11.12±(1.58)	16.27±(1.85)	24.61±(2.16)	32.64±(2.35)	38.49±(2.44)	43.72±(2.49)
	C4.5S	0.96±(0.49)	5.75±(1.17)	11.08±(1.57)	20.67±(2.03)	28.59±(2.27)	34.87±(2.39)	41.12±(2.47)
	C4.5SP	1.84±(0.68)	7.93±(1.35)	13.65±(1.72)	22.96±(2.11)	30.61±(2.31)	36.92±(2.42)	43.37±(2.49)
	AVT-DTLT	1.21±(0.55)	5.86±(1.18)	11.85±(1.62)	21.34±(2.05)	29.14±(2.28)	34.69±(2.39)	42.26±(2.48)
	AVT-DTLTP	2.89±(0.84)	7.94±(1.35)	13.35±(1.70)	23.01±(2.11)	30.17±(2.30)	36.18±(2.41)	43.32±(2.49)
	AVT-DTLY	1.21±(0.55)	3.10±(0.87)	6.32±(1.22)	10.89±(1.56)	20.17±(2.01)	27.11±(2.23)	32.75±(2.35)
	AVT-DTLYP	2.89±(0.84)	3.62±(0.93)	7.38±(1.31)	12.70±(1.67)	21.93±(2.08)	27.69±(2.25)	33.17±(2.36)

that appear close to the roots of the AVTs. Consequently, the partially missing attribute values which correspond to nodes in the AVT that are further away from the root than the attribute tests chosen in the decision tree have no adverse impact on error rate.

AVT-DTL (either with or without pruning) slightly outperforms C4.5 with or without pruning (but no subsetting), with respect to estimated error rates over a broad range of percentages of totally missing attribute values. However, C4.5 with subsetting sometimes yields slightly lower error rates than AVT-DTL. This may be explained by the fact that C4.5 with subsetting is less constrained than AVT-DTL in its choice of tests.

Figure 3.12 compares AVT-DTL with C4.5 variants in terms of the size of the decision trees generated from data with different percentages of partially missing attribute values. AVT-DTL without pruning generates trees that are smaller than those generated by C4.5 (with or without pruning) and C4.5 with subsetting but no pruning. Decision trees generated by AVT-DTL with pruning are comparable to those generated by C4.5 with subsetting and pruning.

3.5 Summary and Discussion

In this chapter, we have presented AVT-DTL, an algorithm for learning decision trees using attribute value taxonomies from partially specified data in which different instances have attribute values specified at different levels of precision. Our approach extends the ontology-based decision tree algorithm proposed in [Zhang et al. (2002)] to learn from, and classify partially specified instances. The technique used in AVT-DTL for handling partially specified attribute values is a generalization of an existing approach to dealing with missing attribute values in decision tree construction and classification.

Experimental results on AVT-DTL have shown that:

- The AVT-DTL algorithm is able to learn robust high accuracy classifiers from data sets consisting of relatively high percentage of partially specified instances.
- The AVT-DTL algorithm yields substantially more compact yet high accuracy decision

trees than standard decision tree learning algorithm (C4.5) and its variants that do not use subsetting or utilize attribute value taxonomies to guide decision tree construction when applied to data sets with fully specified instances as well as data sets with relatively high percentage of missing attribute values.

- The AVT-DTL algorithm achieves a better trade-off between accuracy and complexity of decision tree classifier than the standard decision tree learning algorithm applied to propositionalized data set.
- The AVT-DTL algorithm is more efficient in its use of training data. AVT-DTL can produce classifiers that outperform those by C4.5 using less training samples.

Among the most related previous work, C4.5 with 'subsetting' and RIPPER learner are both able to utilize the so called set-valued attributes. The classification rules constructed by the RIPPER rule learning algorithm proposed by Cohen (1996b) utilize tests for membership in attribute-value sets. C4.5 with 'subsetting' option can also be seen as working with set-valued attributes. However, the attribute value sets considered by these algorithms are not constrained by any AVT (other than the default single level taxonomy with a "don't care" value as the root and primitive values at the leaves). An unconstrained search through candidate subsets of values of each attribute during the learning phase might result in more compact classifiers (e.g. when compactness is measured in terms of the number of nodes in a decision tree) than those produced by AVT-guided learning algorithms such as AVT-DTL. However, in the absence of the structure imposed over sets of attribute values used in constructing the classifier, specifying the outcome of each test (outgoing branch from a node in the decision tree) requires enumerating the members of the set of values corresponding to that branch. Thus, each rule extracted from a decision tree produced by C4.5 with 'subsetting' is a conjunction of disjunctions, making the resulting classifiers difficult to interpret. This is also true of the rules generated by RIPPER using set-valued attributes. In contrast, the rules that utilize abstract attribute values from an AVT are much more compact because each attribute value subset has a name that needs to be specified only once - in the AVT, and the attribute value

sets considered are constrained by the tree structure of the AVT. Consequently, the rules generated from trees produced by AVT-DTL are compact and easy to interpret because each rule is a simple conjunction of conditions of the form (*attribute* = *value*). Because algorithms like RIPPER and C4.5 with 'subsetting' have to search the set of candidate value subsets for each attribute under consideration while adding conditions to a rule or a node to trees, they are computationally more demanding than AVT-DTL. Furthermore, neither C4.5 with 'subsetting' nor RIPPER is equipped to build classifiers from partially specified data. Lastly, many scientific applications require users to be able to explore a given data set using alternative AVTs which reflect different ontological commitments or different ways of conceptualizing a domain. Unlike C4.5 with 'subsetting' and RIPPER, AVT-DTL can utilize a user-supplied AVTs to construct classification rules that are expressed in terms of abstract attribute values specified by the AVTs.

CHAPTER 4. AVT-based Naïve Bayes Learner

In this chapter, we start by reviewing standard Naïve Bayes classifier learning algorithm. We describe in detail AVT-NBL, a natural generalization of the Naïve Bayes learner (NBL), for learning classifiers from AVT and data, including partially specified data. We present experimental results by comparing AVT-NBL with standard Naïve Bayes Learner (NBL) and alternative approach (NBL on propositionalized data) on accuracy, complexity, and robustness of the induced classifiers. At the end of this chapter, we conclude with summary and discussion.

4.1 Naïve Bayes Learner (NBL)

Statistical decision theory provides the foundation for constructing optimal decision rules that minimize risk [Cherkassky and Mulier (1998)]. The Bayes theorem provides such fundamental decision rule. The Bayes theorem calculates the probability of a particular event given some observations. Given the training data D, the *posteriori* probability of a class or hypothesis h, $P(h|D)$ is written as:

$$P(h|D) = \frac{P(h)P(D|h)}{P(D)}$$

$P(h)$ is the estimated *priori* probability of h, and we have $P(h) \geq 0$, $\sum_{h \in H} P(h) = 1$, where H is the hypothesis space. $P(D|h)$ is the *likelihood* of the data D given the hypothesis h. $P(D)$ is called the *evidence*, and is the marginal probability observing the training data D.

The Bayesian learning framework [Mitchell (1997); Duda et al. (2001)] formulates learning as the task of producing a *maximum a posteriori* (MAP) hypothesis. The MAP hypothesis is the hypothesis that has maximum $P(h|D)$:

$$h_{MAP} = \underset{h \in H}{\operatorname{argmax}} P(h|D) = \underset{h \in H}{\operatorname{argmax}} P(h)P(D|h)$$

A Bayesian classifier returns the MAP hypothesis that maximizes the posterior probability $P(h|D)$, which also minimizes the expected loss (or conditional risk). However, unless we have an extremely large data set, there exists the practical difficulty of reliably estimating the conditional joint probability for different instances.

The Naïve Bayes classifier is a simple and yet effective Bayesian classifier that has competitive performance with other more sophisticated classifiers [Langley et al. (1992)]. Naïve Bayes classifier operates under the assumption that each attribute is independent of others given the class. Thus, the joint probability, given a class, can be written as the product of individual class conditional probabilities for each attribute. The Bayesian approach to classifying an instance $X_p = (v_{1p}, v_{2p}, \cdots, v_{Np})$ is to assign it the most probable class $c_{MAP}(X_p)$:

$$\begin{aligned} c_{MAP}(X_p) &= \underset{c_j \in C}{\operatorname{argmax}} P(v_{1p}, v_{2p}, \cdots, v_{Np}|c_j)p(c_j) \\ &= \underset{c_j \in C}{\operatorname{argmax}} p(c_j) \prod_i P(v_{ip}|c_j) \end{aligned}$$

The Bayesian learning framework assumes that data was generated by a parametric model, and that we can use the training data to estimate the model parameters. The task of the Naïve Bayes Learner (NBL) is to estimate $\forall c_j \in C$ and $\forall v_{i_k} \in dom(A_i)$, relevant class probabilities $p(c_j)$ and the class conditional probabilities $P(v_{i_k}|c_j)$ from training data D. These probabilities, which completely specify a Naïve Bayes classifier, can be estimated from a training set D using standard probability estimation methods [Mitchell (1997)] based on relative frequency counts of the corresponding classes and attribute value and class label co-occurrences observed in D. We denote $\sigma_i(v_k|c_j)$ as the frequency count of value v_k of attribute A_i given class label c_j, and $\sigma(c_j)$ as the frequency count of class label c_j in a training set D. Numerical attribute values are handled by assuming that they have "normal" (Gaussian) distribution, and we need to calculate the estimated mean and standard deviation from the training data. Hence, these relative frequency counts for nominal attributes and estimated means and standard variances for numerical attributes completely summarize the information needed for constructing a Naïve Bayes classifier from D, and they constitute *sufficient statistics* for Naïve Bayes learner [Caragea et al. (2004b)]. Following is the pseudo-code for Naïve Bayes classifier learning algorithm.

Naïve Bayes Classifier Learning Algorithm:

Learn-Naïve-Bayes(Examples D)

1. Calculate frequency counts for class labels: $\{\sigma(c_j)|j = 1 \cdots M\}$.
   Calculate frequency counts for each nominal attribute value:
   $\{\forall v_{i_k} \in dom(A_i), \sigma(v_{i_k}|c_j) \,|\, i = 1 \cdots N, j = 1 \cdots M\}$

2. Estimate class probabilities $P(c_j)$ from frequency counts $\{\sigma(c_j)|j = 1 \cdots M\}$:

$$P(c_j) \leftarrow \frac{\sigma(c_j)}{\sum_{c_k \in C} \sigma(c_k)}$$

   Estimate class conditional probabilities $P(v_{i_k}|c_j)$ from frequency counts $\{\forall v_{i_k} \in dom(A_i), \sigma(v_{i_k}|c_j) \,|\, i = 1 \cdots N, j = 1 \cdots M\}$ by using Laplace estimates [Mitchell (1997)]:

$$P(v_{ik}|c_j) \leftarrow \frac{1 + \sigma(v_{i_k}|c_j)}{|D| + \sum_{u_{i_k} \in A_i} \sigma(u_{i_k}|c_j)}$$

3. Estimate mean $\hat{\mu}_i$ and standard deviation $\hat{\sigma}_i$ for numerical attribute A_i from D based on a normal distribution

Classify-Naïve-Bayes(Unlabelled Instance X_p)

- Assign $X_p = (v_{1p}, v_{2p}, \cdots, v_{Np})$ to the most probable class:

$$c_{MAP}(X_p) = \underset{c_j \in C}{\operatorname{argmax}}\, p(c_j) \prod_i P(v_{ip}|c_j)$$

Figure 4.1 General Procedure of Naïve Bayes classifier for learning and classifying.

Despite the fact that the independence assumptions are rarely satisfied in real applications, Naïve Bayes classifier performs very well, and its performance is comparable to that of decision trees and neural networks. Since it only requires a single scan of the data, it scales very well to a very large data set. Naïve Bayes classifier has been successfully used in text classification tasks, such as NewsWeeder [Lang (1995)] and Junk Mail Filter [Sahami et al. (1998)], among others. Friedman and Domingos and Pazzani discussed that even in domains with substantial attribute dependence, it can still achieve optimal classifications under the zero-one loss function [Domingos and Pazzani (1997); Friedman (1997a)].

4.2 AVT-NBL: Algorithm Description

We now introduce AVT-NBL, an algorithm for learning Naïve Bayes classifiers from AVT and data. Given an ordered set of AVTs $T = \{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_N\}$ corresponding to the attributes $A = \{A_1, A_2, \ldots, A_N\}$ and a data set $D = \{(X_p, c_{X_p})\}$ of labeled examples of the form (X_p, c_{X_p}) where $X_p \in I_T$ is a partially or fully specified instance and $c_{X_p} \in C$ is the corresponding class label, the task of AVT-NBL is to construct a Naïve Bayes classifier for assigning X_p to its most probable class $c_{MAP}(X_p)$. As in the case of NBL, we assume that each attribute is independent of the other attributes given the class.

Let $\Gamma = \{\gamma_1, \gamma_2, \cdots, \gamma_N\}$ be a global cut where, γ_i stands for a cut through $\mathcal{T}_i$. A Naïve Bayes classifier defined on the instance space I_Γ is completely specified by a set of class conditional probabilities for each value of each attribute. Suppose we denote the table of class conditional probabilities associated with values in γ_i by $CPT(\gamma_i)$. Then the Naive Bayes classifier defined over the instance space I_Γ is specified by $h(\Gamma) = \{CPT(\gamma_1), CPT(\gamma_2), \cdots, CPT(\gamma_N)\}$.

If each cut $\gamma_i \in \Gamma_0$ is chosen to correspond to the primitive values of the respective attribute i.e., $\forall i\, \gamma_i = Leaves(\mathcal{T}_i)$, $h(\Gamma_0)$ is simply the standard Naïve Bayes Classifier based on the attributes $A_1, A_2, \cdots, A_N$. If each cut $\gamma_i \in \Gamma$ is chosen to pass through the root of each AVT, i.e., $\forall i\, \gamma_i = \{Root(\mathcal{T}_i)\}$, $h(\Gamma)$ simply assigns each instance to the class that is a priori most probable.

AVT-NBL starts with the Naïve Bayes Classifier that is based on the most abstract value

of each attribute (the most general hypothesis in H_T) and successively refines the classifier (hypothesis) using a criterion that is designed to trade off between the accuracy of classification and the complexity of the resulting Naïve Bayes classifier. Successive refinements of Γ correspond to an ordering of Naive Bayes classifiers based on the structure of the AVTs in T.

For example, in Figure 4.2, $\hat{\Gamma}$ is a cut refinement of Γ, and hence the corresponding hypothesis $h(\hat{\Gamma})$ is a refinement of $h(\Gamma)$. Relative to the two cuts, Table 4.1 shows the conditional probability tables that we need to compute during learning for $h(\Gamma)$ and $h(\hat{\Gamma})$ respectively (assuming $C = \{+, -\}$ as two possible class labels). From the class conditional probability table, we can count the number of class conditional probabilities needed to specify the corresponding Naïve Bayes classifier. As shown in Table 4.1, the total number of class conditional probabilities for $h(\Gamma)$ and $h(\hat{\Gamma})$ are 8 and 10 respectively.

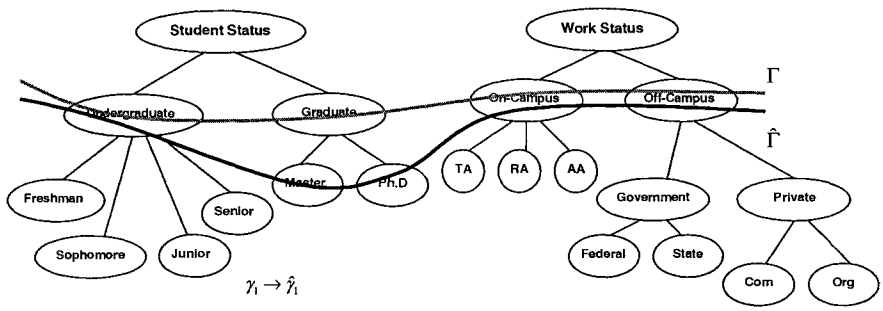


Figure 4.2 Cut refinement.

4.2.1 Class Conditional Frequency Counts

Similar to the computation of frequency counts on AVT that has been described in Chapter 3, we can calculate class conditional frequency counts on AVT. Given an attribute value taxonomy $\mathcal{T}_i$ for attribute A_i, we define a tree of class conditional frequency counts $CCFC(\mathcal{T}_i)$ such that there is a one-to-one correspondence between the nodes of the AVT $\mathcal{T}_i$ and the nodes of the corresponding $CCFC(\mathcal{T}_i)$. It follows that the class conditional frequency counts associated with a non leaf node of $CCFC(\mathcal{T}_i)$ should correspond to the aggregation of the corresponding

Table 4.1 Conditional probability tables. This table shows the entries of conditional probability tables associated with two global cut Γ and $\hat{\Gamma}$ shown in Figure 4.2 (assuming C={+,-}).

CPT for $h(\Gamma)$		
Value	Pr(+)	Pr(−)
Undergraduate	*P(Undergraduate|+)*	*P(Undergraduate|−)*
Graduate	*P(Graduate|+)*	*P(Graduate|−)*
On-Campus	*P(On-Campus|+)*	*P(On-Campus|−)*
Off-Campus	*P(Off-Campus|+)*	*P(Off-Campus|−)*
CPT for $h(\hat{\Gamma})$		
Value	Pr(+)	Pr(−)
Undergraduate	*P(Undergraduate|+)*	*P(Undergraduate|−)*
Master	*P(Master|+)*	*P(Master|−)*
Ph.D	*P(Ph.D|+)*	*P(Ph.D|−)*
On-Campus	*P(On-Campus|+)*	*P(On-Campus|−)*
Off-Campus	*P(Off-Campus|+)*	*P(Off-Campus|−)*

class conditional frequency counts associated with its children. Because each cut through an AVT $\mathcal{T}_i$ corresponds to a partition of the set of possible values in $Nodes(\mathcal{T}_i)$ of the attribute A_i, the corresponding cut γ_i through $CCFC(\mathcal{T}_i)$ specifies a valid class conditional probability table $CPT(\gamma_i)$ for the attribute A_i.

In the case of numerical attributes, AVTs are defined over intervals based on observed values for the attribute in the data set. Each cut through the AVT corresponds to a partition of the numerical attribute into a set of intervals. We calculate the class conditional probabilities for each interval. As in the case of nominal attributes, we define a tree of class conditional frequency counts $CCFC(\mathcal{T}_i)$ for each numerical attribute A_i. $CCFC(\mathcal{T}_i)$ is used to calculate the conditional probability table $CPT(\gamma_i)$ corresponding to a cut γ_i.

When all of the instances in the data set D are fully specified, estimation of $CCFC(\mathcal{T}_i)$ for each attribute is straightforward: we simply estimate the class conditional frequency counts associated with each of the primitive values of A_i from the data set D and use them recursively to compute the class conditional frequency counts associated with the non-leaf nodes of $CCFC(\mathcal{T}_i)$.

When some of the data are partially specified, we can use a 2-step process for computing $CCFC(\mathcal{T}_i)$: First we make an upward pass aggregating the class conditional frequency counts based on the specified attribute values in the data set; Then we propagate the counts associated with partially specified attribute values down through the tree, augmenting the counts at lower levels according to the distribution of values along the branches based on the subset of the data for which the corresponding values are fully specified. This procedure can be seen as a special case of EM (Expectation Maximization) algorithm [Dempster et al. (1977)] to estimate sufficient statistics for $CCFC(\mathcal{T}_i)$.

Let $\sigma_i(v|c_j)$ be the frequency count of value v of attribute A_i given class label c_j in a training set D, and $p_i(v|c_j)$ the estimated class conditional probability of value v of attribute A_i given class label c_j in a training set D. Let $\pi(v, \mathcal{T}_i)$ be the set of all children (direct descendants) of a node with value v in $\mathcal{T}_i$; $\Lambda(v, \mathcal{T}_i)$ the list of ancestors, including the root, for v in $\mathcal{T}_i$. The procedure of computing $CCFC(\mathcal{T}_i)$ is shown below.

Input: Training data D and $T = \{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_N\}$

Output: $CCFC(\mathcal{T}_1), CCFC(\mathcal{T}_2), \ldots, CCFC(\mathcal{T}_N)$.

1. Calculate frequency counts $\sigma_i(v|c_j)$ for each node v in $\mathcal{T}_i$ using the class conditional frequency counts associated with the specified values of attribute A_i in training set D.
2. For each attribute value v in $\mathcal{T}_i$ which received non-zero counts as a result of step 1, aggregate the counts upward from each such node v to its ancestors $\Lambda(v, \mathcal{T}_i)$: $\sigma_i(w|c_j)_{w \in \Lambda(v,\mathcal{T}_i)} \leftarrow \sigma_i(w|c_j) + \sigma_i(v|c_j)$.
3. Starting from the root, recursively propagate the counts corresponding to partially specified instances at each node v downward according to the observed distribution among its children to obtain updated counts for each child $u_l \in \pi(v, \mathcal{T}_i)$:

$$\sigma_i(u_l|c_j) = \begin{cases} \left(\frac{\sigma_i(v|c_j)}{|\pi(v,\mathcal{T}_i)|}\right) & \text{If } \sum_{k=1}^{|\pi(v,\mathcal{T}_i)|} \sigma_i(u_k|c_j)=0, \\ \sigma_i(u_l|c_j)\left(1 + \frac{\sigma_i(v|c_j) - \sum_{k=1}^{|\pi(v,\mathcal{T}_i)|} \sigma_i(u_k|c_j)}{\sum_{k=1}^{|\pi(v,\mathcal{T}_i)|} \sigma_i(u_k|c_j)}\right) & \text{Otherwise.} \end{cases}$$

Figure 4.3 Computation of Class Conditional Frequency Counts on AVT

Now that we have estimated the class conditional frequency counts for all attribute value taxonomies, we can calculate the conditional probability table with regard to any global cut Γ. Let $\Gamma = \{\gamma_1, \cdots, \gamma_N\}$ be a global cut where γ_i stands for a cut through $CCFC(T_i)$. The estimated conditional probability table $CPT(\gamma_i)$ associated with the cut γ_i can be calculated from $CCFC(T_i)$ using Laplace estimates [Mitchell (1997); Kohavi et al. (1997)].

$$p_i(v|c_j)_{v \in \gamma_i} \leftarrow \frac{1/|D| + \sigma_i(v|c_j)}{|\gamma_i|/|D| + \sum_{u \in \gamma_i} \sigma_i(u|c_j)}$$

Recall that the Naïve Bayes Classifier $h(\Gamma)$ based on a chosen global cut Γ is completely specified by the conditional probability tables associated with the cuts in Γ: $h(\Gamma) = \{CPT(\gamma_1), \cdots, CPT(\gamma_N)\}$.

4.2.2 MDL Principle Applied to AVT-NBL

The scoring function that we use to evaluate a candidate AVT-guided refinement of a Naïve Bayes Classifier is based on a variant of the minimum description length (MDL) score [Rissanen (1978)] which provides a basis for trading off the complexity against the error of the model. An overly complex model with many parameters is likely to result in poor parameter estimations with high variance (or overfitting) because of a limited number of training examples or a large fraction of partially specified values. A simpler model, if it has the right level of abstraction, is likely to yield more reliable estimates of the parameters and to be a better classifier.

In 1965, Kolmogorov first introduced the notion of *algorithmic complexity* for characterization of randomness of a data set [Kolmogorov (1965)]. He defined the algorithmic complexity of a data set to be the shortest binary code describing the data. Rissanen (1978) proposed using Kolmogorov's characterization of randomness as a tool for inductive inference, and this is known as the MDL principle. The MDL principle can be naturally viewed as a Bayesian MAP (maximum a posteriori) estimator [Mitchell (1997)], and is derived from the Bayes theorem [Wallace and Boulton (1968)] and information theoretical considerations [Shannon (1948)]. In intuitive terms, the goal of a learner is to compress information from the training data and encode it in the form of a classifier. Communicating information in a training set can be

modelled by a two-part message that encodes the model (or classifier) h and the data D given the model h.

The length of this message is written as $L(D,h) = L(h) + L(D|h)$, with $L(h)$ to be the length of the message (measured by the number of bits) for specification of the model and $L(D|h)$ to be the length of the message (measured by the number of bits) for specification of the data given the model. The data dependent term $L(D|h)$ can further be divided into a parameter block $L(w^*|h)$ with the best fit parameters w^* of the model, and a data block $L(D|w^*,h)$ corresponding to the bits needed to send the data using those parameters. Assuming optimal encoding is used for each component of the message, we can express $L(D,h)$ as follows [Wallace and Freeman, 1987]:

$$L(D,h) = L(h) + L(D|h) = -\log P(h) - \log P(D|h) = L(h) + L(w^*|h) + L(D|w^*,h)$$

When we assume equal priors $P(h)$, the tradeoff between alternative models in a class of models (hypothesis class H) is captured by the data dependent term. Simpler models will have a more compact parameter block, but a longer message to fit the data. As the number of parameters increase, the data block will decrease in size. Bayesian approach to learning prescribes finding a classifier $h \in H$ that minimizes the total length of the message.

Friedman et al. (1997) suggested the use of a conditional MDL (CMDL) score in the case of hypotheses that are used for classification (as opposed to modelling the joint probability distribution of a set of random variables) to capture this tradeoff. In general Bayesian network settings, let B be a Bayesian network and D be a training set. The MDL scoring function can be written as:

$$MDL(B|D) = \left(\frac{\log|D|}{2}\right)|B| - LL(B|D)$$

where $|B|$ is the number of parameters in the network. The first term describes the length for specification of the network B, and the second term is the negation of the *log likelihood* of B given D. Given the representation of labelled training samples $\{< X_p, c_{X_p} > | p = 1 \cdots |D|\}$, where $X_p = (v_{1p}, v_{2p}, \cdots, v_{Np})$, the log likelihood function can be extended to:

$$LL(B|D) = \sum_{p=1}^{|D|} \log P_B(c_{X_p}|v_{1p}, v_{2p}, \cdots, v_{Np}) + \sum_{p=1}^{|D|} \log P_B(v_{1p}, v_{2p}, \cdots, v_{Np})$$

Friedman et al. (1997) argued that only the first term is related to the score of the network as a classifier, and directly related to its predictive accuracy. While the second term sometimes becomes the dominant term when there are many attributes, a relatively large error in the conditional term may not be reflected in the MDL score, and this may result in a poor classifier. Therefore, Friedman et al. (1997) suggested using *conditional log likelihood* score (i.e., $CLL(B|D) = \sum_{p=1}^{|D|} \log P_B(c_{X_p}|v_{1p}, v_{2p}, \cdots, v_{Np})$) to specialize the scoring function to the classification task. And accordingly, the MDL score has been revised to the CMDL score.

In general, computation of the CMDL score is not feasible for Bayesian networks with an arbitrary structure [Friedman et al. (1997)]. However, in the case of Naive Bayes classifiers induced by a set of AVT, as shown below, it is possible to efficiently calculate the CMDL score. We also replace B with h, the corresponding Naïve Bayes classifier with regard to a chosen global cut.

$$CMDL(h|D) = \left(\frac{\log|D|}{2}\right) size(h) - CLL(h|D)$$
$$where,\ CLL(h|D) = |D| \sum_{p=1}^{|D|} \log P_h(c_{X_p}|v_{1p}, \cdots, v_{Np})$$

Here, $P_h(c_{X_p}|v_{1p}, \cdots, v_{Np})$ denotes the conditional probability assigned to the class $c_{X_p} \in C$ associated with the training sample $X_p = (v_{1p}, v_{2p}, \cdots, v_{Np})$ by the classifier h, $size(h)$ is the number of parameters used by h, $|D|$ the size of the data set, and $CLL(h|D)$ is the conditional log likelihood of the hypothesis h given the data D. In the case of a Naïve Bayes classifier h, $size(h)$ corresponds to the total number of class conditional probabilities needed to describe h. Because each attribute is assumed to be independent of the others given the class in a Naïve Bayes classifier, we have:

$$CLL(h|D) = |D| \sum_{p=1}^{|D|} \log\left(\frac{P(c_{X_p}) \prod_i P_h(v_{ip}|c_{X_p})}{\sum_{j=1}^{|C|} P(c_j) \prod_i P_h(v_{ip}|c_j)}\right)$$

where $P(c_j)$ is the prior probability of the class c_j which can be estimated from the observed class distribution in the data D.

4.2.3 Searching for a Compact Naïve Bayes Classifier

Because each attribute is assumed to be independent of the others given the class, the search for the AVT-based Naïve Bayes classifier (AVT-NBC) can be performed efficiently by optimizing the criterion independently for each attribute. This results in a hypothesis h that intuitively trades off the complexity of Naïve Bayes classifier (in terms of the number of parameters used to describe the relevant class conditional probabilities) against the accuracy of classification. The algorithm terminates when none of the candidate refinements of the classifier yield statistically significant improvement in the CMDL score. The procedure is outlined in Figure 4.4.

Input: Training data D and $CCFC(\mathcal{T}_1), CCFC(\mathcal{T}_2), \ldots, CCFC(\mathcal{T}_N)$.

Output: A Naïve Bayes classifier trading off the complexity against the error.

1. Initialize each γ_i in $\Gamma = \{\gamma_1, \gamma_2, \cdots, \gamma_N\}$ to $\{Root(\mathcal{T}_i)\}$.
2. Estimate probabilities that specify the hypothesis $h(\Gamma)$.
3. For each cut γ_i in $\Gamma = \{\gamma_1, \gamma_2, \cdots, \gamma_N\}$:
   A. Set $\delta_i \leftarrow \gamma_i$
   B. Until there are no updates to γ_i
      i. For each $v \in \delta_i$
         a. Generate a refinement γ_i^v of γ_i by replacing v with $\pi(v, \mathcal{T}_i)$, and refine Γ accordingly to obtain $\hat{\Gamma}$. Construct corresponding hypothesis $h(\hat{\Gamma})$
         b. If $CMDL(h(\hat{\Gamma})|D) < CMDL(h(\Gamma)|D)$, replace Γ with $\hat{\Gamma}$ and γ_i with γ_i^v
      ii. $\delta_i \leftarrow \gamma_i$
4. Output $h(\Gamma)$

Figure 4.4 Searching for compact AVT-based Naïve Bayes classifier

4.2.4 Handling Partially Specified Data in AVT-NBL

Like AVT-DTL, AVT-NBL also has an embedded mechanism to handle partially specified data.

- In the calculation of class conditional frequency counts (CCFC) for AVTs, we take into account the case when some of the data are partially specified. We use a two-step procedure to estimate the frequency counts which can be seen as a special case of EM (Expectation Maximization) algorithm under the assumption that the attributes are independent given the class. The class conditional probabilities with regard to attribute values (fully specified or partially specified) in any cut in AVT can be calculated by doing normalization on the cut.

- In the calculation of the conditional likelihood $CLL(h|D)$, we distinguish between two cases when D contains partially specified instances:

  (1) When a partially specified value of attribute A_i for an instance lies on the cut γ through $CCFC(\mathcal{T}_i)$ or corresponds to one of the descendants of the nodes in the cut. In this case, we can treat that instance as though it were fully specified relative to the Naïve Bayes classifier based on the cut γ of $CCFC(\mathcal{T}_i)$ and use the class conditional probabilities associated with the cut γ to calculate its contribution to $CLL(h|D)$;

  (2) When a partially specified value (say v) of A_i is an ancestor of a subset (say λ) of the nodes in γ. In this case, $p(v|c_j) = \sum_{u_i \in \lambda} p(u_i|c_j)$, such that we can aggregate the class conditional probabilities of the nodes in λ to calculate the contribution of the corresponding instance to $CLL(h|D)$.

AVT-NBL can directly handle partially specified data during both the learning phase and classification phase. We do not need to do any data preprocessing on partially specified data.

4.3 Alternative Approaches

As we have described in Section 3.3, there are two alternative approaches in learning classifiers from data and attribute value taxonomies. One is the approach that treats partially specified attribute values as if they were totally missing, and one is the AVT-based propositionalization method.

We call the resulting algorithm - NBL applied to AVT-based propositionalized version of the data - Prop-NBL. Note that the Boolean features created by the propositionalization technique described above are not independent given the class. A Boolean attribute that corresponds to any node in an AVT is necessarily correlated with Boolean attributes that correspond to its descendants as well as its ancestors in the tree. Therefore, the statistical dependence among the Boolean attributes in the propositionalized representation of the original data set can degrade the performance of the Naïve Bayes classifier that relies on the independence of attributes given class.

Against this background, we experimentally compare AVT-NBL with Prop-NBL and the standard Naïve Bayes algorithm (NBL).

4.4 Performance Study

Our experiments were designed to explore the performance of AVT-NBL relative to that of NBL and PROP-NBL. The performance of the algorithms was evaluated with respect to complexity, generalization, and robustness of the induced classifiers.

4.4.1 Experiments

We select 37 data sets from the UCI Data Repository [1], among which 8 data sets use only nominal attributes and 29 data sets have both nominal attributes and numerical attributes. Every numerical attribute in the 29 data sets has been discretized into a maximum of 10 bins. For the data sets without expert-generated AVTs, the AVTs on both nominal and numer-

[1] http://www.ics.uci.edu/~mlearn/MLRepository.html

ical attributes were generated using AVT-Learner, a Hierarchical Agglomerative Clustering algorithm to construct AVTs for learning [Kang et al. (2004)].

We have designed the following three sets of experiments. In each case, the error rate and the size (as measured by the number of class conditional probabilities used to specify the learned classifier) were estimated using 10-fold cross-validation, and we calculate 90% confidence interval on the error rate.

- The first set of experiments compares the performance of AVT-NBL, NBL, and PROP-NBL on the original data.

- The second set of experiments explores the performance of the algorithms on data sets with different percentages of totally missing and partially missing attribute values. Three data sets with a pre-specified percentage (10%, 30%, or 50%, *excluding* the missing values in the original data set) of totally or partially missing attribute values were generated by assuming that the missing values are uniformly distributed on the nominal attributes (refer to Section 3.4.1.2 for details).

- The third set of experiments were designed to investigate the performance of classifiers generated by AVT-NBL, Prop-NBL, and NBL as a function of the training set size. We divided each data set into two disjoint parts: a training pool and a test pool. Training sets of different sizes, corresponding to 10%, 20%, ..., 100% of the training pool, were sampled and used to train Naïve Bayes classifier using AVT-NBL, Prop-NBL, and NBL. The resulting classifiers were evaluated on the entire test pool. The experiment was repeated 9 times for each training set size. The entire process was repeated using 3 different random partitions of data into training and test pools. The accuracy of the learned classifiers on the examples in the test pool were averaged across the $9 \times 3 = 27$ runs.

Table 4.2 Comparison of error rate and size of classifiers generated by NBL, PROP-NBL and AVT-NBL on 37 UCI benchmark data. The error rates and the sizes were estimated using 10- fold cross validation. We calculate 90% confidence interval on the error rates. The size of the classifiers for each data set is constant for NBL and Prop-NBL, and for AVT-NBL, the size shown represents the average across the 10-cross validation experiments.

DATA SET	NBL		PROP-NBL		AVT-NBL	
	ERROR	*SIZE*	ERROR	*SIZE*	ERROR	*SIZE*
Anneal	6.01(±1.30)	*954*	10.69(±1.69)	*2886*	1.00(±0.55)	*666*
Audiology	26.55(±5.31)	*3696*	27.87(±5.39)	*8184*	23.01(±5.06)	*3600*
Autos	22.44(±4.78)	*1477*	21.46(±4.70)	*5187*	13.17(±3.87)	*805*
Balance-Scale	8.64(±1.84)	*63*	11.52(±2.09)	*195*	8.64(±1.84)	*60*
Breast-Cancer	28.32(±4.82)	*84*	27.27(±4.76)	*338*	27.62(±4.78)	*62*
Breast-w	2.72(±1.01)	*180*	2.86(±1.03)	*642*	2.72(±1.01)	*74*
Car	14.47(±1.53)	*88*	15.45(±1.57)	*244*	13.83(±1.50)	*80*
Colic	17.93(±3.28)	*252*	20.11(±3.43)	*826*	16.58(±3.18)	*164*
Credit-a	14.06(±2.17)	*204*	18.70(±2.43)	*690*	13.48(±2.13)	*124*
Credit-g	24.50(±2.23)	*202*	26.20(±2.28)	*642*	24.60(±2.23)	*154*
Dermatology	2.18(±1.38)	*876*	1.91(±1.29)	*2790*	2.18(±1.38)	*576*
Diabetes	22.53(±2.47)	*162*	25.65(±2.58)	*578*	22.01(±2.45)	*108*
Glass	22.90(±4.71)	*637*	28.04(±5.04)	*2275*	19.16(±4.41)	*385*
Heart-c	14.19(±3.29)	*370*	16.50(±3.50)	*1205*	12.87(±3.16)	*210*
Heart-h	13.61(±3.28)	*355*	14.97(±3.41)	*1155*	13.61(±3.28)	*215*
Heart-Statlog	16.30(±3.69)	*148*	16.30(±3.69)	*482*	13.33(±3.39)	*78*
Hepatitis	10.97(±4.12)	*174*	9.03(±3.78)	*538*	7.10(±3.38)	*112*
Hypothyroid	4.32(±0.54)	*436*	6.68(±0.67)	*1276*	4.22(±0.54)	*344*
Ionosphere	7.98(±2.37)	*648*	8.26(±2.41)	*2318*	5.41(±1.98)	*310*
Iris	4.00(±2.62)	*123*	4.67(±2.82)	*435*	5.33(±3.01)	*90*
Kr-vs-kp	12.11(±0.95)	*150*	12.20(±0.95)	*306*	12.08(±0.95)	*146*
Labor	8.77(±6.14)	*170*	10.53(±6.67)	*546*	10.53(±6.67)	*70*
Letter	27.17(±0.52)	*4186*	34.40(±0.55)	*15002*	29.47(±0.53)	*2652*
Lymph	14.19(±4.70)	*240*	18.24(±5.21)	*660*	15.54(±4.88)	*184*
Mushroom	4.43(±1.30)	*252*	4.45 (±1.30)	*682*	0.14(±0.14)	*202*
Nursery	9.67(±1.48)	*135*	10.59 (±1.54)	*355*	9.67(±1.48)	*125*
Primary-Tumor	49.85(±4.45)	*836*	52.51(±4.45)	*1782*	52.21(±4.45)	*814*
Segment	10.91(±1.06)	*1183*	11.86(±1.10)	*4193*	10.00(±1.02)	*560*
Sick	2.52(±0.42)	*218*	4.51(±0.55)	*638*	2.17(±0.39)	*190*
Sonar	0.96(±1.11)	*1202*	0.96(±1.11)	*4322*	0.48(±0.79)	*312*
Soybean	7.03 (±1.60)	*1900*	8.19(±1.72)	*4959*	5.71(±1.45)	*1729*
Splice	4.64(±0.61)	*864*	4.08(±0.57)	*2727*	4.23(±0.58)	*723*
Vehicle	33.33(±2.66)	*724*	32.98(±2.65)	*2596*	32.15(±2.63)	*368*
Vote	9.89(±2.35)	*66*	9.89(±2.35)	*130*	9.89(±2.35)	*64*
Vowel	64.24(±2.50)	*1320*	63.33(±2.51)	*4675*	57.58(±2.58)	*1122*
Waveform-5000	35.96(±1.11)	*1203*	36.38(±1.12)	*4323*	34.92(±1.11)	*825*
Zoo	6.93(±4.57)	*259*	5.94(±4.25)	*567*	3.96(±3.51)	*245*

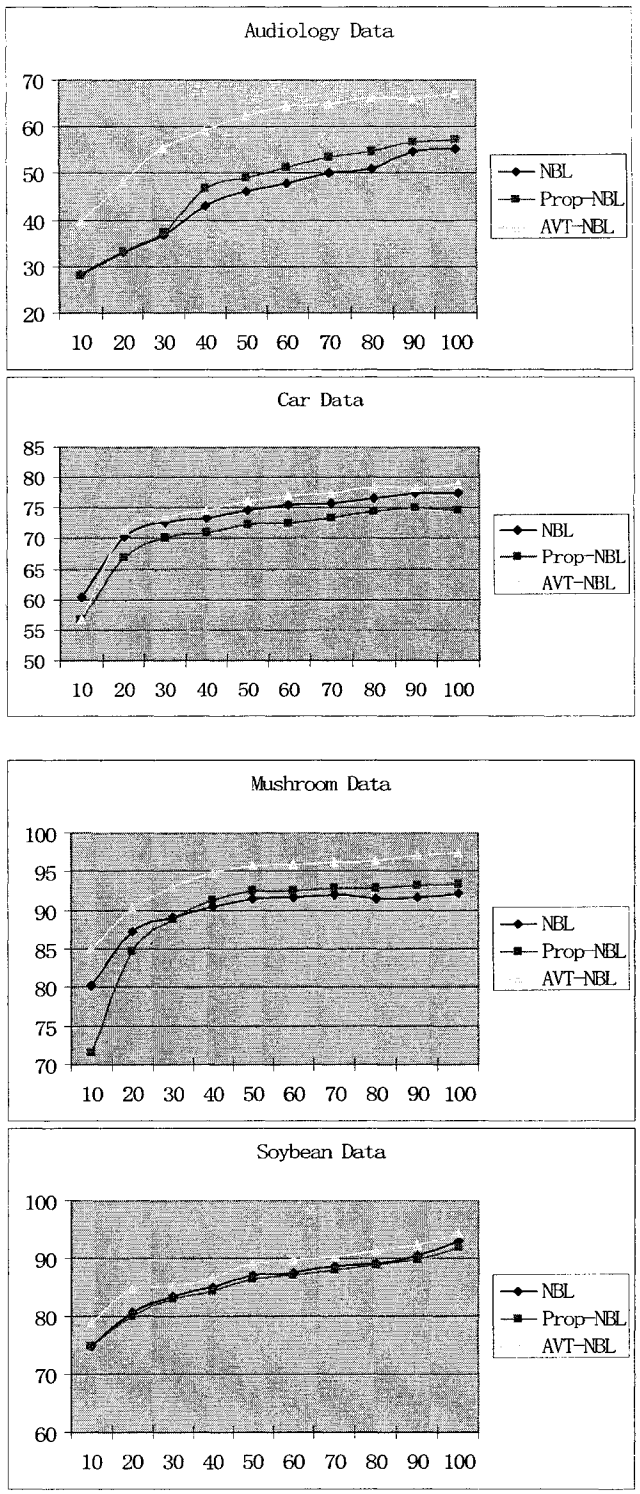


Figure 4.5 Classifier accuracy as a function of training set size on several data sets by AVT-NBL, Prop-NBL, and NBL respectively. Note that the X axis shows the percentage of training instances that has been sampled in training the Naïve Bayes classifier, and Y axis shows the predictive accuracy in percentage.

Table 4.3 Comparison of error rates on data with 10%, 30% and 50% partially or totally missing values. The error rates were estimated using 10-fold cross validation, and we calculate 90% confidence interval on each error rate.

DATA		PARTIALLY MISSING			TOTALLY MISSING		
METHODS		NBL	PROP-NBL	AVT-NBL	NBL	PROP-NBL	AVT-NBL
MUSHROOM	10%	4.65(±1.33)	4.69(±1.34)	0.30(±0.30)	4.65(±1.33)	4.76(±1.35)	1.29(±071)
	30%	5.28 (±1.41)	4.84(±1.36)	0.64(±0.50)	5.28 (±1.41)	5.37(±1.43)	2.78(±1.04)
	50%	6.63(±1.57)	5.82(±1.48)	1.24(±0.70)	6.63(±1.57)	6.98(±1.61)	4.61(±1.33)
NURSERY	10%	15.27(±1.81)	15.50(±1.82)	12.85(±1.67)	15.27(±1.81)	16.53(±1.86)	13.24(±1.70)
	30%	26.84(±2.23)	26.25(±2.21)	21.19(±2.05)	26.84(±2.23)	27.65(±2.24)	22.48(±2.09)
	50%	36.96(±2.43)	35.88(±2.41)	29.34(±2.29)	36.96(±2.43)	38.66(±2.45)	32.51(±2.35)
SOYBEAN	10%	8.76(±1.76)	9.08(±1.79)	6.75(±1.57)	8.76(±1.76)	9.09(±1.79)	6.88(±1.58)
	30%	12.45(±2.07)	11.54(±2.00)	10.32(±1.90)	12.45(±2.07)	12.31(±2.05)	10.41(±1.91)
	50%	19.39(±2.47)	16.91(±2.34)	16.93(±2.34)	19.39 (±2.47)	19.59(±2.48)	17.97(±2.40)

4.4.2 Results

We summarize below the results of experiments that compared the performance of standard NBL with that of AVT-NBL as well as the standard learning algorithms applied to a propositionalized version of the data set (PROP-NBL) [Zhang and Honavar (2003)].

4.4.2.1 AVT-NBL yields lower error rates than NBL and PROP-NBL on the original fully specified data

Table 4.2 shows the estimated error rates of the classifiers generated by the AVT-NBL, NBL, and PROP-NBL on 37 UCI benchmark data sets. According to the results, the error rate of AVT-NBL is substantially smaller than that of NBL and PROP-NBL. It is worth noting that PROP-NBL (NBL applied to a transformed data set using Boolean features that correspond to nodes of the AVTs) generally produces classifiers that have higher error rates than NBL. This can be explained by the fact that the Boolean features generated from an AVT are generally not independent given the class.

4.4.2.2 AVT-NBL yields classifiers that are substantially more compact than those generated by PROP-NBL and NBL

The shaded columns in Table 4.2 compare the total number of class conditional probabilities needed to specify the classifiers produced by AVT-NBL, NBL, and PROP-NBL on original data. The results show that AVT-NBL is effective in exploiting the information supplied by the AVT to generate accurate yet compact classifiers. Thus, AVT-guided learning algorithms offer an approach to compressing class conditional probability distributions that is different from the statistical independence-based factorization used in Bayesian Networks.

4.4.2.3 AVT-NBL yields significantly lower error rates than NBL and PROP-NBL on partially specified data and data with totally missing values

Table 4.3 compares the estimated error rates of AVT-NBL with that of NBL and PROP-NBL in the presence of varying percentages (10%, 30% and 50%) of partially missing attribute values and totally missing attribute values. Naïve Bayes classifiers generated by AVT-NBL have substantially lower error rates than those generated by NBL and PROP-NBL, with the differences being more pronounced at higher percentages of partially (or totally) missing attribute values.

4.4.2.4 AVT-NBL produces more accurate classifiers than NBL and Prop-NBL for a given training set size

Figure 4.5 shows the plot of the accuracy of the classifiers learned as a function of training set size for *Audiology* data, *Car* data, *Mushroom* data, and *Soybean* data. We obtained similar results on other benchmark data sets used in this study. Thus, AVT-NBL is *more efficient* than NBL and Prop-NBL in its use of training data.

4.5 Summary and Discussion

In this chapter, we have described AVT-NBL [2], an algorithm for learning Naïve Bayes classifiers from attribute value taxonomies (AVT) and data in which different instances may have attribute values specified at different levels of abstraction. AVT-NBL is a natural generalization of the standard algorithm for learning Naïve Bayes Classifiers. The standard Naïve Bayes learner (NBL) can be viewed as a special case of AVT-NBL by collapsing a multi-level AVT associated with each attribute into a corresponding single level AVT whose leaves correspond to the primitive values of the attribute.

Our experimental results presented in this chapter show that:

- AVT-NBL is able to learn substantially compact and more accurate classifiers on a broad range of data sets than those produced by standard NBL and Prop-NBL (applying NBL to data with an augmented set of Boolean attributes).
- When applied to data sets in which attribute values are partially specified or totally missing, AVT-NBL can yields classifiers that are more accurate and compact than those generated by NBL and Prop-NBL.
- AVT-NBL is more efficient in its use of training data. AVT-NBL produces classifiers that outperform those produced by NBL using substantially fewer training examples.

Thus, AVT-NBL offers an effective approach to learning compact (hence more comprehensible) accurate classifiers from data - including data that are *partially specified.* AVT-guided learning algorithms offer a promising approach to knowledge acquisition from autonomous, semantically heterogeneous information sources, where domain specific AVTs are often available and data are often partially specified. Details on learning classifiers from semantically heterogeneous data sources are discussed in Chapter 6.

[2] A Java implementation of AVT-NBL and the data sets and AVTs used in this study are available at: http://www.cs.iastate.edu/~jzhang/ICDM04/index.html

CHAPTER 5. General Framework For Learning Concise and Accurate Classifiers from Attribute Value Taxonomies and Data

We describe in the following section a general framework for designing algorithms to learn classifiers from AVT-extended data sources. Base on this framework, we demonstrate our approach by showing that AVT-DTL and AVT-NBL presented in the previous two chapters can be instantiated using this general framework.

5.1 AVT-Based Classifier Learners: A General Framework

There are essentially three elements in learning classifiers from AVT-extended data sources: (1) A procedure for identifying estimated sufficient statistics on AVTs from data; (2) A procedure for building and refining hypothesis; (3) A performance criterion for making a tradeoff between complexity and accuracy of the generated classifiers. In what follows, we discuss each element in details.

5.1.1 Identifying Estimated Sufficient Statistics

Building a classifier only needs certain *statistics*. A statistic is simply a function of the data D. For example, the sample mean and standard deviation are known as sufficient statistics for a Gaussian distribution. Other statistics include counts of instances with specified attribute values, the most frequent value of an attribute, etc. In building a Naïve Bayes classifier, we only need the frequency counts of the corresponding classes and the frequency counts of attribute values given class labels.

Definition 5.1 (SUFFICIENT STATISTIC) [Casella and Berger (2001)] *A statistic $\mathcal{S}(D)$ is called a* sufficient statistic *for a parameter θ if $\mathcal{S}(D)$ (loosely speaking) provides all the in-*

formation needed for estimating the parameter θ from data D. A sufficient statistic s for a parameter θ is called a minimal sufficient statistic *if for every sufficient statistic s' for θ there exists a function g_s such that $g_s(s'(D)) = s(D)$.*

Knowledge of the values of sufficient statistic is all that is required to build a classifier. Although the original data set D is a trivial sufficient statistic for learning the hypothesis h using L applied to D, we are mostly interested in sufficient statistics that are minimal or substantially smaller in size than the whole data set D.

We can generalize the notion of a sufficient statistic for a parameter θ to yield the notion of a sufficient statistic $\mathcal{S}_L(D, h)$ for learning a hypothesis h using a learning algorithm L applied to a data set D.

Definition 5.2 (SUFFICIENT STATISTIC FOR A LEARNING ALGORITHM) [Caragea et al. (2004b)] *We say that $\mathcal{S}_L(D, h)$ is a sufficient statistic for learning the hypothesis h using a learning algorithm L if there exists an algorithm that accepts $\mathcal{S}_L(D, h)$ as input and outputs h.*

For many learning algorithms, sufficient statistics are frequency counts or class conditional frequency counts for attribute values. Given a hierarchical structured AVT, we can define a tree of frequency counts or class conditional frequency counts as the sufficient statistics for the AVT-Based classifier learning algorithms. More specifically, with regard to an attribute value taxonomy $\mathcal{T}_i$ for attribute A_i, we define a tree of frequency counts $Counts(\mathcal{T}_i)$ and a tree of class conditional frequency counts $CCFC(\mathcal{T}_i)$. Detailed algorithms on calculating the tree of frequency counts or class conditional frequency counts on AVT have been depicted in Figure 3.3 and Figure 4.3.

Briefly, if all the instances are fully specified in the AVT-extended data source, the (class conditional) frequency counts associated with a non leaf node should correspond to the aggregation of the corresponding (class conditional) frequency counts associated with its children. $Counts(\mathcal{T}_i)$ or $CCFC(\mathcal{T}_i)$ can be computed in one upward pass. In the case where some of the data are partially specified in AVT-extended data source, we can use a 2-step process for computing $Counts(\mathcal{T}_i)$ or $CCFC(\mathcal{T}_i)$ [Zhang and Honavar (2004a)]. This procedure can be

seen as a special case of EM (Expectation Maximization) algorithm [Dempster et al. (1977)] to estimate sufficient statistics for $Counts(\mathcal{T}_i)$ or $CCFC(\mathcal{T}_i)$ under the assumption that the attributes are independent given the class.

5.1.2 Building and Refining Hypothesis

As we have mentioned earlier, for a particular global cut Γ there is a corresponding hypothesis class H_Γ, and we can learn a hypothesis $h(\theta|\Gamma)$ with parameters θ from this hypothesis class H_Γ using a learning algorithm L. The total number of global cuts $|\Delta|$ grows exponentially with the scale of AVTs, and so does the number of possible hypotheses. Since an exhaustive search over the complete hypothesis space $\{H_\Gamma|\Gamma \in \Delta\}$ is computationally infeasible, we need a strategy to search through the resulting hypothesis space.

Recall that a cut $\hat{\gamma}_i$ is a refinement of a cut γ_i if $\hat{\gamma}_i$ is obtained by replacing at least one attribute value $v \in \gamma_i$ by its descendant attribute values. A global cut $\hat{\Gamma}$ is a refinement of a global cut Γ if at least one cut in $\hat{\Gamma}$ is a refinement of a cut in Γ. When $\hat{\Gamma}$ is a cut refinement of Γ, the corresponding hypothesis $h(\hat{\Gamma})$ is a *hypothesis refinement* of $h(\Gamma)$. The sufficient statistics for hypothesis refinement is defined as follows.

Definition 5.3 (Sufficient Statistics for Hypothesis Refinement) [Caragea et al. (2004b)] *We denote $\mathcal{S}_L(D, h_i \rightarrow h_{i+1})$ as the sufficient statistic for hypothesis refinement from h_i to h_{i+1}, if the learner L accepts h_i and a sufficient statistic $\mathcal{S}_L(D, h_i \rightarrow h_{i+1})$ as inputs and outputs an updated hypothesis h_{i+1}.*

Based on gathered sufficient statistics, our goal is to search for the optimal hypothesis $h(\Gamma^*)$ from $\{H_\Gamma|\Gamma \in \Delta\}$, where Γ^* is an optimal level of abstraction (i.e., an optimal cut) that is decided by the learning algorithm L using certain performance measurement P.

We use a top-down refinement on the global cut to greedily explore the design space of the corresponding classifier. Hypothesis refinements in AVT-based learning are conducted through cut refinements in AVTs. Our general strategy is to start by building a classifier that is based on the most abstract global cut and successively refine the classifier (hypothesis) by cut refinement. Therefore, the learning algorithm L generates a sequence of cut refinements $\Gamma_0, \Gamma_1, \cdots, \Gamma^*$,

which corresponds to a sequence of hypothesis refinements $h(\Gamma_0), h(\Gamma_1), \cdots, h(\Gamma^*)$, until a final optimal cut Γ^* and an optimal classifier $h(\Gamma^*)$ is obtained.

5.1.3 Trading off the Complexity against the Error

For almost every learning algorithm L, there is a performance measurement P that is explicitly or implicitly optimized by L. For example, some performance measurements include predictive accuracy, statistical significance tests, and many other information criteria. However, the lack of a good performance measurement causes the learning algorithm to build an overly-complex model as the classifier which shows excellent performance on training data but poor performance on test data. This problem of overfitting is a general problem that many learning algorithms seek to overcome.

Of particular interest to us are those criteria that can make tradeoffs between the accuracy and the complexity of the model [Akaike (1974); Rissanen (1978)], thereby having a built-in mechanism to overcome overfitting. By making this tradeoff, we are able to learn classifiers that are both compact and accurate.

For example, the Minimum Description Length (MDL) principle [Rissanen (1978)] compresses the training data D and encodes it by a hypothesis h such that it minimizes the length of the message that encodes both h and the data D given h.

In order to perform hypothesis refinements effectively, we need a performance criterion P that can decide if we need to make a refinement from $h(\Gamma)$ to $h(\hat{\Gamma})$. Also this criterion should be able to decide whether we should stop making refinements and output a final hypothesis as the classifier. Different learning algorithms may use different performance criteria, and thus may have different formats and expressions of refinement sufficient statistics.

By combining the three elements of AVT-based classifier learners, we can write the procedure in Figure 5.1 to show this general learning framework.

1. Identify estimated sufficient statistics $\mathcal{S}_L(D)$ for AVTs as counts $\{CCFC(\mathcal{T}_i) \mid i = 1, ..., N\}$ or $\{FC(\mathcal{T}_i) | i = 1, ..., N\}$.
2. Initialize the global cut Γ to the most abstract cut Γ_0.
3. Based on the estimated sufficient statistic, generate a hypothesis $h(\Gamma)$ corresponding to the current global cut Γ and learn its parameters.
4. Generate a cut refinement $\hat{\Gamma}$ on Γ, and construct hypothesis $h(\hat{\Gamma})$
5. Calculate $\mathcal{S}_L(D, h(\Gamma) \rightarrow h(\hat{\Gamma}))$ for hypothesis refinement from $h(\Gamma)$ to $h(\hat{\Gamma})$.
6. Based on performance criterion P, if stopping criterion is met, then output $h(\Gamma)$ as the final classifier; else if the condition for hypothesis refinement is met, set current hypothesis to $h(\hat{\Gamma})$ by replacing Γ with $\hat{\Gamma}$, else keep $h(\Gamma)$, and goto step 4;

Figure 5.1 A General Learning Framework for AVT-Based Classifiers

5.2 Two Instantiations of AVT-Based Classifier Learners

Based on our description of the general learning framework for AVT-Based Classifier Learners, we can identify corresponding elements for AVT-NBL and AVT-DTL. We can see AVT-NBL and AVT-DTL as the instantiations of this general AVT-based classifier learning framework.

5.2.1 AVT-Based Naïve Bayes Learner (AVT-NBL)

As we have described extensively in Chapter 4, AVT-NBL [Zhang and Honavar (2004a)] is an extension of the standard Naïve Bayes learning algorithm that effectively exploits user-supplied AVTs to construct compact and accurate Naïve Bayes classifier from partially specified data. We can easily identify the three elements in the learning framework for AVT-NBL as follows:

(1) The sufficient statistics $\mathcal{S}_L(D)$ for AVT-NBL is the class conditional frequency counts $\{CCFC(\mathcal{T}_i) | i = 1, ..., N\}$.

(2) The hypothesis refinements strictly follow the procedure of cut refinements in the frame-

work. When a global cut Γ is specified, there is a corresponding Naïve Bayes classifier $h(\Gamma)$ that is completely specified by a set of class conditional probabilities for the attribute values on Γ. Because each attribute is assumed to be independent of others given the class, the search for the AVT-based Naïve Bayes classifier (AVT-NBC) can be performed efficiently by optimizing the criterion independently for each attribute.

(3) The performance criterion that AVT-NBL optimizes is the Conditional Minimum Description Length (CMDL) score suggested by Friedman et al. (1997). CMDL score can be calculated as follows:

$$CMDL(h(\Gamma)|D) = \left(\frac{\log|D|}{2}\right) size(h(\Gamma)) - CLL(h(\Gamma)|D)$$
$$where,\ CLL(h(\Gamma)|D) = |D|\sum_{p=1}^{|D|} \log P_h(c_{X_p}|v_{1p}, \cdots, v_{Np})$$

where, $P_h(c_{X_p}|v_{1p}, \cdots, v_{Np})$ is the class conditional probability, $size(h(\Gamma))$ is the number of parameters used by $h(\Gamma)$, $|D|$ the size of the data set, and $CLL(h(\Gamma)|D)$ is the conditional log likelihood of the hypothesis $h(\Gamma)$ given the data D. In the case of a Naïve Bayes classifier, $size(h(\Gamma))$ corresponds to the total number of class conditional probabilities needed to describe $h(\Gamma)$.

The sufficient statistics for hypothesis refinement in AVT-NBL can be quantified by the difference between their respective CMDL scores: $s_L(D, h(\Gamma) \rightarrow h(\hat{\Gamma})) = CMDL(h(\hat{\Gamma})|D) - CMDL(h(\Gamma)|D)$. If $s_L(D, h(\Gamma) \rightarrow h(\hat{\Gamma})) > 0$, $h(\Gamma)$ is refined to $h(\hat{\Gamma})$. This refinement procedure terminates when no further refinement can make improvement in the CMDL score (i.e., the stoping criterion).

5.2.2 AVT-Based Decision Tree Learner (AVT-DTL)

AVT-DTL [Zhang and Honavar (2003)] implements a top-down AVT-guided search in the decision tree hypothesis space, and is able to learn a compact and accurate decision tree classifier from partially specified data. Similarly, we can identify the three elements in the learning framework for AVT-DTL as follows:

(1) The sufficient statistics $\mathcal{S}_L(D)$ for AVT-DTL is the frequency counts $\{FC(\mathcal{T}_i)|i = 1, ..., N\}$.

(2) The hypothesis refinement is incorporated into the process of decision tree construction. The cut refinement is done by keeping track of "pointing vectors" (as shown in Figure 3.5) in the AVTs. Each "pointing vector" is a set of pointers, and each pointer points to a values in an AVT.

The union of the set of pointing vectors at all leaves of a partially constructed decision tree corresponds to a global cut in AVTs. Obviously, any global cut in the constructed decision tree has a corresponding global cut in AVTs. At each stage of decision tree construction, we have a current set of pointing vectors as the global cut Γ being explored, and a corresponding partially constructed decision tree to be the hypothesis $h(\Gamma)$. AVT-DTL indirectly makes refinement on Γ by updating each pointing vector, and hence makes hypothesis refinement on $h(\Gamma)$ and grows the decision tree accordingly. AVT-DTL does not have the independent assumption on attributes given the class, the search is conducted globally to make refinements on possible cuts.

(3) The performance criterion that AVT-DTL uses is the standard information gain or gain ratio [Quinlan (1993)]. The sufficient statistic for hypothesis refinement is exactly the information criterion: $s_L(D, h(\Gamma) \rightarrow h(\hat{\Gamma})) = info(\Gamma \rightarrow \hat{\Gamma})$, where $info(\Gamma \rightarrow \hat{\Gamma})$ is the information gain (or gain ratio) when current decision tree $h(\Gamma)$ has been extended to $h(\hat{\Gamma})$. The stopping criterion for AVT-DTL is the same for standard decision tree. For example, such stopping criterion can be χ^2 test to test statistical significance on further split.

5.3 Summary and Discussion

Ontology-aware classifier learning algorithms are needed to explore data from multiple points of view, and to understand the interaction between data and knowledge. By exploiting ontologies in the form of attribute value taxonomies in learning classifiers from data, we are able to construct robust, accurate and easy-to-comprehend classifiers within a particular domain of interest.

In this chapter, we have described a general framework for deriving ontology-aware algorithms for learning classifiers from data when ontologies take the form of attribute value

taxonomies. We illustrate the application of this framework in the case of AVT-based variants of decision tree and Naïve Bayes classifiers. However, this framework can be used to derive AVT-based variants of other learning algorithms. For our future work, it would be interesting to design AVT-based variants of algorithms for construction Bag-of-words classifiers, Bayesian Networks, Nonlinear Regression Classifiers, and Hyperplane classifiers (Perceptron, Winnow Perceptron, and Support Vector Machines).

CHAPTER 6. Learning Concise and Accurate Classifiers from Semantically Heterogeneous Data

In this chapter, we precisely define the problem of learning classifiers from distributed, semantically heterogeneous data sources viewed from a learner's perspective. We extend our previous approach for learning Naïve Bayes classifier from semantically homogeneous data with associated attribute value taxonomies and partially specified data (described in details in chapter 4) to an approach for learning compact and accurate Naïve Bayes classifier from distributed and heterogeneous data sources. We present a principled way to reduce the problem of learning from semantically heterogeneous data to the problem of learning from distributed partially specified data by reconciling semantic heterogeneity using AVT-mappings, and describe a sufficient statistics based solution. We prove that the resulting algorithm is exact relative to its centralized counterpart. We provide experimental results on synthesized distributed and semantically heterogeneous data sets. Our experiment results verified our theoretical analysis on the exactness of resulting algorithm.

6.1 Distributed and Ontology Extended Data Sources

6.1.1 Ontology-extended data sources

Recall that in Chapter 2 we have defined distributed data sources and semantically heterogeneous data sources. We assume data D for learning are distributed over $D_1, D_2, \ldots, D_K$. Each data source D_i contains only a fragment of the whole data D, and we consider two common types of data fragmentation [Caragea et al. (2004b)]: *horizontal fragmentation*, wherein subsets of data tuples are stored at different locations; and *vertical fragmentation*, wherein sub-tuples of data tuples are stored at different locations. The nature of distributed data

sources usually prohibit shipping raw data from each of the sites to a centralized location for learning.

Furthermore, each individual distributed data source has its own associated ontology (i.e., a collection of names for concepts and relation types organized in partial ordering by the type subtype relation [Sowa (1999)]). Different data sources may have different data semantics by following different ontologies. Such a data source for learning is the so called semantically heterogeneous data. We formally define ontology-extended data sources to describe such data as follows.

Let D_i be a data set associated with the ith data source, described by the set of attributes $\{A_1^i, \cdots, A_n^i\}$ and $O_i = \{\Lambda_1^i, \cdots, \Lambda_n^i\}$ a simple ontology associated with this data set. The element $\Lambda_j^i \in O_i$ corresponds to the attribute A_j^i and describes the type of that particular attribute. The type of an attribute can be a standard type (e.g., real value or string) or a hierarchical type. A hierarchical type is defined as an ordering of a set of terms (e.g., the values of an attribute) [Bonatti et al. (2003)]. Of special interest to us are *attribute value taxonomies* (AVT) as shown in Figure 6.1 and Figure 6.2 (These are the same AVTs shown in Chapter 2, for easy reference we reprint the figures again).

The schema S_i of a data source D_i is given by the set of attributes $\{A_1^i, \cdots, A_n^i\}$ used to describe the data together with their respective types $\{\Lambda_1^i, \cdots, \Lambda_n^i\}$ described by the ontology O_i, i.e., $S_i = \{A_1^i : \Lambda_1^i, \cdots, A_n^i : \Lambda_n^i\}$. An *ontology-extended data source* [Caragea et al. (2004a)] is defined as a tuple $\mathcal{D}_i = <D_i, S_i, O_i>$, where D_i is the actual data in the data source, S_i is the schema of the data source and O_i is the ontology associated with the data source. Obviously, the following condition needs to be satisfied: $D_i \subseteq \Lambda_1^i \times \cdots \times \Lambda_n^i$, which means that each attribute A_j^i can take values in the set Λ_j^i defined in the ontology O_i.

For simplicity, we use $[D_i, \mathbf{\Lambda}_i]$ to represent the distributed data source $\mathcal{D}_i$ with associated AVT $\mathbf{\Lambda}_i$. The entire data sources can be represented in the form $([D_1, \mathbf{\Lambda}_1], \cdots, [D_K, \mathbf{\Lambda}_K])$.

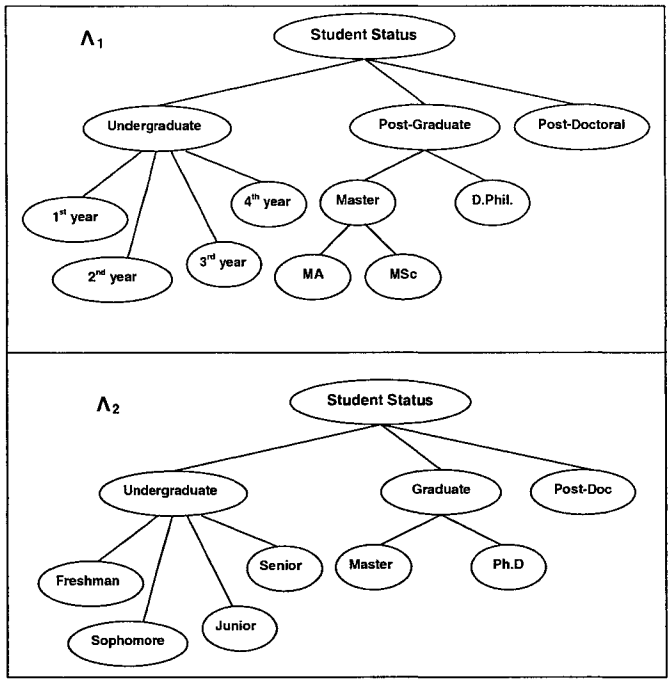


Figure 6.1 AVTs for data source D_1 and D_2 respectively

6.1.2 Learner Perspective and Integrable Ontologies

When learning from semantically heterogeneous data where each data source has an associated ontology that is different from the user ontology (or learner ontology), we need a principled approach to integrating ontologies and data from semantically heterogeneous data sources. Therefore, one important requirement for successful learning from heterogeneous data is to reconcile the semantic heterogeneity.

Let $[D_1, \mathbf{\Lambda}_1], \cdots, [D_K, \mathbf{\Lambda}_K]$ be an ordered set of K ontology-extended data sources and L a learner that poses queries against these heterogeneous data sources. We introduce the notion of *interoperation constraints* and define a *learner perspective* to reflect the learner perspective of integrating a set of data source AVTs according to the learner AVT. Inspired from a model of ontology-extended relational algebra [Bonatti et al. (2003)] and a general model of ontology-extended data sources [Caragea et al. (2004a)], we focus our discussion on the integration of AVTs.

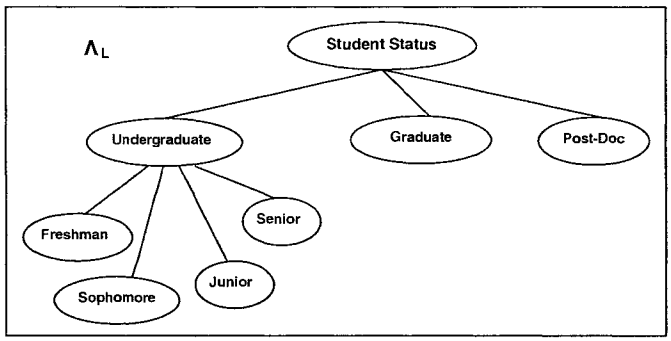


Figure 6.2 The Learner AVT

Definition 6.1 (INTEROPERATION CONSTRAINTS) [Bonatti et al. (2003); Caragea et al. (2004a)] *Let $\mathbf{\Lambda}_1, \cdots, \mathbf{\Lambda}_K$ be K different attribute value taxonomies, and let $\mathbf{\Lambda}_L$ be a learner attribute value taxonomy. Interoperation constraints IC is a set of constraints that exist between elements from $\mathbf{\Lambda}_i$ and elements from $\mathbf{\Lambda}_L$ in the following formats:*

- $x : \mathbf{\Lambda}_i \equiv y : \mathbf{\Lambda}_L$ *(x is semantically equivalent to y)*
- $x : \mathbf{\Lambda}_i \preceq y : \mathbf{\Lambda}_L$ *(x is semantically below y)*
- $x : \mathbf{\Lambda}_i \succeq y : \mathbf{\Lambda}_L$ *(x is semantically above y)*

Using the same example, Figure 6.2 gives a learner AVT and Figure 6.1 shows two data source AVTs associated with D_1 and D_2. We have the following interoperation constraints, among others: $Undergraduate : \mathbf{\Lambda}_1 \equiv Undergraduate : \mathbf{\Lambda}_L$, $1^{st}year : \mathbf{\Lambda}_1 \equiv Freshman : \mathbf{\Lambda}_L$, $Post\text{-}graduate : \mathbf{\Lambda}_2 \equiv Graduate : \mathbf{\Lambda}_L$, $D.Phil : \mathbf{\Lambda}_2 \equiv Ph.D : \mathbf{\Lambda}_L$, $MA : \mathbf{\Lambda}_1 \preceq Master : \mathbf{\Lambda}_L$, $MSc : \mathbf{\Lambda}_1 \preceq Graduate : \mathbf{\Lambda}_L$, etc.

Definition 6.2 (LEARNER PERSPECTIVE) [Caragea et al. (2004a)] *A learner perspective $LP(\mathbf{\Lambda}_L, IC)$ is defined by a learner attribute value taxonomy $\mathbf{\Lambda}_L$ and a set of interoperation constraints IC defined between $\mathbf{\Lambda}_1, \cdots, \mathbf{\Lambda}_K$ and $\mathbf{\Lambda}_L$.*

To integrate a set of data source AVTs from a learner perspective, we need to find a set of mappings from the elements in data source AVTs to the elements in learner AVT.

Definition 6.3 (INTEGRABLE AVTs) [Caragea et al. (2004a)] *Under a learner perspective* $LP(\mathbf{\Lambda}_L, IC)$ *and corresponding interoperation constraints* IC*, a set of data source attribute value taxonomies* $\mathbf{\Lambda}_1, \cdots, \mathbf{\Lambda}_K$ *is integrable according to learner* $\mathbf{\Lambda}_L$*, if there exist* K *injective partial mappings* $\phi_1, \cdots, \phi_K$ *from* $\mathbf{\Lambda}_1, \cdots, \mathbf{\Lambda}_K$ *respectively to* $\mathbf{\Lambda}_L$ *with the following two properties:*

- *For all* $x, y \in \mathbf{\Lambda}_i$*, if* $x \prec_i y$ *then* $\phi_i(x) \prec \phi_i(y)$ *(order preservation);*
- *For all* $x \in \mathbf{\Lambda}_i$ *and* $y \in \mathbf{\Lambda}_L$*, if* $(x : \mathbf{\Lambda}_i \ op \ y : \mathbf{\Lambda}_L) \in IC$*, then* $\phi_i(x)$ *op* y *in* $\mathbf{\Lambda}_L$ *(interoperation constraints preservation).*

It is trivial to convert a learner perspective $LP(\mathbf{\Lambda}_L, IC)$ with interoperation constraints into a set of explicit mappings [Caragea et al. (2004a)]. By either exact name matchings or equality constraints mappings, we are able to find all valid mappings in the form $term_1 : \mathbf{\Lambda}_i \rightarrow term_2 : \mathbf{\Lambda}_L$.

Definition 6.4 (AVT-MAPPINGS) *Based on interoperation constraints, we define an AVT-mapping* $M(\mathbf{\Lambda}_i, \mathbf{\Lambda}_L)$ *between a data source AVT* $\mathbf{\Lambda}_i$ *and the learner AVT* $\mathbf{\Lambda}_L$ *by enumerating all partial injective mappings in the form "*$term_1 : \mathbf{\Lambda}_i \rightarrow term_2 : \mathbf{\Lambda}_L$*". Similarly, we define entire AVT-mappings as a set* $\{M(\mathbf{\Lambda}_i, \mathbf{\Lambda}_L) | i = 1, \cdots, K\}$.

Following the above example with its interoperation constraints, we obtain corresponding *AVT-mappings* in Table 6.1.

To be able to answer learner's queries on sufficient statistics from heterogeneous data sources, we use the AVT-mappings to resolve the semantic differences between the learner AVT and data source AVTs. Recall that we use $\mathcal{S}_L(D, h)$ to describe the sufficient statistic answered by data source D. If we have a distributed data source $[D_i, \mathbf{\Lambda}_i]$, and its corresponding AVT-mapping $M(\mathbf{\Lambda}_i, \mathbf{\Lambda}_L)$, then the sufficient statistic answered by this data source can be denoted as $\mathcal{S}_L(([D_i, \mathbf{\Lambda}_i] | M(\mathbf{\Lambda}_i, \mathbf{\Lambda}_L)), h)$. Here, we use the notion $([D_i, \mathbf{\Lambda}_i] | M(\mathbf{\Lambda}_i, \mathbf{\Lambda}_L))$ to represent the data source $[D_i, \mathbf{\Lambda}_i]$ from the learner's perspective, virtually a data set with same semantics as the learner's. Therefore, when AVT-mappings are properly applied, the semantic heterogeneity of the disparate data sources becomes transparent to the learner.

Table 6.1 AVT-mappings from $\mathbf{\Lambda}_1$ to $\mathbf{\Lambda}_L$.

$M(\mathbf{\Lambda}_1, \mathbf{\Lambda}_L)$
$Undergraduate \rightarrow Undergraduate$
$1^{st}year \rightarrow Freshman$
$2^{nd}year \rightarrow Sophomore$
$3^{rd}year \rightarrow Junior$
$4^{th}year \rightarrow Senior$
$Post\text{-}Graduate \rightarrow Graduate$
$Master \rightarrow Master$
$D.Phil \rightarrow Ph.D$
$Post\text{-}Doctoral \rightarrow Post\text{-}Doc$

6.1.3 Complete Data from a Learner Perspective

In a horizontally fragmented distributed setting, let $[D_1, \mathbf{\Lambda}_1], \cdots, [D_K, \mathbf{\Lambda}_K]$ be K a set of K ontology-extended data sources with respective attribute value taxonomies, and let $\mathbf{\Lambda}_L$ be a learner attribute value taxonomy (user ontology). The complete virtual data set D in a centralized location can be seen as the multi-set union (with possible duplicate instances) of the available data sources $[D_1, \mathbf{\Lambda}_1], \cdots, [D_K, \mathbf{\Lambda}_K]$. This is done by AVT-mappings from each attribute values in a distributed source to the corresponding value in the learner AVT. Thus, $D = ([D_1, \mathbf{\Lambda}_1] | M(\mathbf{\Lambda}_1, \mathbf{\Lambda}_L)) \cup \cdots \cup ([D_K, \mathbf{\Lambda}_K] | M(\mathbf{\Lambda}_K, \mathbf{\Lambda}_L))$.

In a vertically fragmented distributed setting, a individual data source only has a subset of the attributes used to describe the data. In this case, we can assume that there is a unique index number associated with each instance, so that we can assemble the instances of D from the corresponding instance fragments stored in $D_1 \cdots D_K$ with appropriate AVT-mappings.

We are interested in producing an algorithm $L_{\aleph}$ for learning from distributed and semantically heterogeneous data relative to its centralized counterpart L for learning from centralized data. In order to evaluate the effectiveness and quality of $L_{\aleph}$, we use exactness [Caragea et al. (2004b)] as a criterion to compare the classifier generated by $L_{\aleph}$ with that produced by its centralized counterpart L. We say $L_{\aleph}$ is *exact* with respect to its centralized counterpart L if the hypothesis produced by $L_{\aleph}$ is identical to that produced by L from the integrated data D

obtained by appropriately integrating the data source $[D_1, \mathbf{\Lambda}_1], \cdots, [D_K, \mathbf{\Lambda}_K]$.

6.1.4 Distributed Partially Specified Data

One direct consequence of information integration from semantically heterogeneous sources is to generate partially specified data [Zhang and Honavar (2003)]). The reason is that when we integrate data from a distributed data source by corresponding AVT-mappings, it is very likely that instances are specified at different levels of precision with respect to the explicitly defined learner AVT $\mathbf{\Lambda}_L$. For example, if the learner AVT describes data at a lower level of abstraction than one or more data source AVTs, the resulting data set D is *partially* specified from the learner's perspective.

A cut γ_i induces a partition on the set of primitive values in $\mathcal{T}_i$, and $\Gamma = \{\gamma_1, \gamma_2, ..., \gamma_N\}$ defines a global cut through $\mathbf{\Lambda}$. Any Γ also specifies a level of abstraction that reflects the user's perspective for the domain. For some users, the level of abstraction can be fixed to reflect the user's special interest on the domain. While for other users, the learning algorithms decide an optimal level of abstraction (i.e., a optimal cut Γ^*) to build the classifiers. In this case, the learning algorithm is able to search for Γ^* based on some performance criterion (e.g., a tradeoff between accuracy and complexity of the classifier associated with Γ^* [Rissanen (1978)]).

With respect to a certain level of abstraction (i.e., a cut Γ in $\mathbf{\Lambda}_L$, expressed as $\Gamma(\mathbf{\Lambda}_L)$), we can define distributed fully specified instance and distributed partially specified instance accordingly as follows. $\Gamma(\mathbf{\Lambda}_L)$ also defines the level of abstraction at which the user queries are formulated.

Definition 6.5 (DISTRIBUTED FULLY/PARTIALLY SPECIFIED INSTANCE) *If Γ represents current level of abstraction in learner AVT (either specified by a particular user or by the learning algorithm), $X_p = (v_{1p}, v_{2p}, ..., v_{Np})$ is an instance from data source D_q, and $v_{ip|M(\mathbf{\Lambda}_q, \mathbf{\Lambda}_L)}$ returns the correspondent of v_{ip} according to the AVT-mappings, then X_p is:*

- *Fully specified with respect to Γ, if $\forall i$, $v_{ip|M(\mathbf{\Lambda}_q, \mathbf{\Lambda}_L)}$ is on or below the cut Γ in $\mathbf{\Lambda}_L$.*
- *Partially specified with respect to Γ, if $\exists v_{ip} \in X_p$, $v_{ip|M(\mathbf{\Lambda}_q, \mathbf{\Lambda}_L)}$ is above the cut Γ in $\mathbf{\Lambda}_L$.*

A learner's level of abstraction $\Gamma(\mathbf{\Lambda}_L)$ determines a level of abstraction $\Gamma(\mathbf{\Lambda}_q)$ in each distributed data source D_q (by applying the corresponding mappings). When correspondent attribute value $v_{ip|M(\mathbf{\Lambda}_q,\mathbf{\Lambda}_L)}$ is below the specified cut Γ, it is fully specified because there is always a corresponding value on the cut that can replace the current value from the perspective of the current level of abstraction. However, when $v_{ip|M(\mathbf{\Lambda}_q,\mathbf{\Lambda}_L)}$ is above the cut, there are several descendant values of $v_{ip|M(\mathbf{\Lambda}_q,\mathbf{\Lambda}_L)}$ on the cut. It is uncertain which value will be its actual attribute value, and hence partially specified.

A correspondent attribute value can switch between being a fully specified value and being a partially specified value when the level of abstraction changes. For example, in Figure 6.2, when the current cut is Undergraduate, Graduate, Post-Doc, then an attribute value "Post-Graduate" in D_1 would be fully specified, because its correspondent "Graduate" is on the cut. When the cut changes to Undergraduate, Master, Ph.D, Post-Doc, this attribute value "Post-Graduate" in D_1 becomes partially specified, because further details about 'Post-Graduate" in D_1 is unknown. Similarly, an instance from distributed data sources can be fully specified or partially specified depending on the current level of abstraction.

6.2 Learning Classifiers from Semantically Heterogeneous Data

6.2.1 Problem Description

The problem of learning classifiers from semantically heterogeneous data (i.e., ontology-extended data sources) can be formulated as follows: Given a collection of ontology-extended data sources $[D_1, \mathbf{\Lambda}_1], \cdots, [D_K, \mathbf{\Lambda}_K]$, a learner perspective $LP(\mathbf{\Lambda}_L, IC)$ which implies a set of mappings $\{M(\mathbf{\Lambda}_i, \mathbf{\Lambda}_L) | i = 1, \cdots, K\}$, a set of constraints Z on distributed data, a hypothesis class H and a performance criterion P, the task of the learner L_d is to output a hypothesis $h \in H$ that optimizes P using only operations allowed by Z (e.g., the prohibition of shipping raw data from each data site). Our goal is to design algorithms that can learn from such semantically heterogeneous data sources and is able to find the best predictive models according to criteria that take into account both the complexity and the accuracy of the model.

In our learning scenario, $L_{\aleph}$ is the algorithm that is able to learn from semantically het-

erogeneous data without shipping raw data, and L is the corresponding algorithm that learns from centralized data. We are interested in designing algorithm L_R that is *exact* relative to its centralized counterpart L.

6.2.2 Sufficient Statistics Based Solution

Our goal is to design algorithms for learning compact and accurate classifiers from distributed, semantically heterogeneous data sources. Our general strategy is based on a method of transforming algorithms for learning classifiers from data into algorithms for learning classifiers from distributed, semantically heterogeneous data [Caragea et al. (2004a)], and the approach of AVT-based classifier learners for learning compact and accurate classifiers from attribute value taxonomies and data [Zhang and Honavar (2003); Zhang and Honavar (2004a); Zhang et al. (2005b)].

We can extend our general framework on AVT-based classifier learners to handle semantically heterogeneous data. This learning task also has three elements: (1) the element of identifying estimated sufficient statistics from semantically heterogeneous data sources; (2) the element of building and refining hypothesis; and (3) the element of trading off the complexity against the error.

The element of identifying sufficient statistics from semantically heterogeneous data involves a procedure for specifying the information needed for learning as a *query* and a procedure for answering this query from distributed data. The procedure for answering queries from distributed data entails the decomposition of a posed query into sub-queries that the individual data sources can answer, followed by the composition of the partial answers into a final answer to the initial query. If the distributed data sources are also semantically heterogeneous, mappings between the data sources ontologies and a user ontology need to be applied in the process of query answering in order to reconcile the semantical differences [Caragea et al. (2004a)]. The exactness of the solution depends on the correctness of the procedure for query decomposition and answer composition. More specifically, we consider information extraction from distributed data by decomposing each statistical query Q posed by the learner into sub-

queries $q_1, \cdots, q_K$ that can be answered by the individual data sources $[D_1, \Lambda_1], \cdots, [D_K, \Lambda_K]$ respectively, and a procedure for combining the answers to the sub-queries into an answer to the original query Q. Therefore, we solve semantic heterogeneity by information integration of data sources from a learner perspective, and reduce the problem to learning classifiers from distributed partially specified data. Our learning paradigm can be depicted by Figure 6.3.

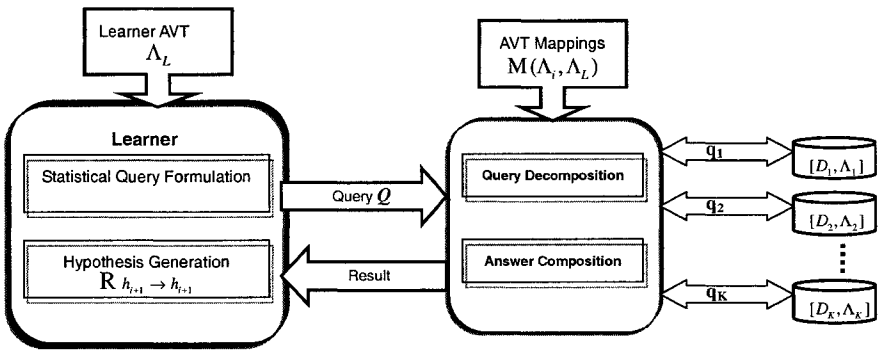


Figure 6.3 Learning from Distributed, Semantically Heterogeneous Data

This is a general learning framework for learning compact and accurate classifier from semantically heterogeneous data, and it can be applied to a large class of learning algorithms (e.g., Naive Bayes, Decision Trees, Support Vector Machines, etc.). All the information needed for constructing the classifier from data can be obtained using a suitable set of statistical queries from the data. We illustrate the application of this approach in the case of learning Naive Bayes classifiers from distributed, semantically heterogeneous data. Based on our previous work on AVT-NBL [Zhang and Honavar (2004a)], we will show how to transform it into an algorithm for learning compact and accurate Naive Bayes classifiers from distributed, semantically heterogeneous data sources.

6.2.3 Sufficient Statistics for AVT-NBL

As we have described in chapter 4 and section 5.2.1 in chapter 5, AVT-NBL is a natural generalization of the standard Naïve Bayes learner (NBL) for learning classifiers from user-supplied AVT and data, including data that are partially specified. The basic idea of AVT-NBL is to start with the Naïve Bayes Classifier that is based on the most abstract attribute

values in AVTs and successively refine the classifier based on a scoring function – a Conditional Minimum Description Length (CMDL) score suggested by Friedman et al. (1997) to make a tradeoff between the accuracy of classification and the complexity of the resulting classifier.

The sufficient statistics for AVT-NBL can be written as $\mathcal{S}_L(D, h)$. First, we need to calculate a tree of class conditional frequency counts (CCFC) that is bonded by the given AVTs. Given $\mathcal{T}_i$ for A_i, we calculate class conditional frequency counts $\mathcal{CCFC}_i$ (as a simplification of $CCFC(\mathcal{T}_i)$) such that there is a one-to-one correspondence between the nodes of the $\mathcal{T}_i$ and the nodes of the corresponding $\mathcal{CCFC}_i$. Therefore, the initial queries of the learner are on CCFC for all attributes, and $\{\mathcal{CCFC}_1, \ldots, \mathcal{CCFC}_N\}$ is the first set of sufficient statistics. The second set of sufficient statistics is an ordered sequence of Conditional Minimum Description Length (CMDL) scores calculated during hypothesis refinements for the purpose of finding the optimal cut (see chapter 4 for the formulae of calculating CMDL score). Therefore, the subsequent statistical queries are posed by the learner on the calculations of a series of CMDL scores. Because each CMDL score is calculated with respect to a corresponding cut Γ through $\mathbf{\Lambda}$, we denote the sequence by $\{\mathcal{CMDL}_{\Gamma^1}, \mathcal{CMDL}_{\Gamma^2}, \cdots\}$, where $\mathcal{CMDL}_{\Gamma^i}$ denote the CMDL score with regard to the cut Γ^i. Hence, the sufficient statistics for AVT-NBL can be identified as follows.

Sufficient Statistics $\mathcal{S}_L(D, h)$ for AVT-NBL

- Frequency Counts for Class Labels:

  $\{\sigma(c_j)|j = 1 \cdots M\}$

- Class Frequency Frequency Counts for $\mathbf{\Lambda}$:

  $\{\mathcal{CCFC}_i|\, i = 1 \cdots N\}$

- Conditional Minimum Description Length Scores:

  $\{\mathcal{CMDL}_{\Gamma^1}, \mathcal{CMDL}_{\Gamma^2}, \cdots\}$

Since there is a one to one correspondence between a global cut Γ and a Naïve Bayes classifier $h(\Gamma)$, the hypothesis refinement on $h(\Gamma)$ is done by a refinement of current cut Γ by

the guidance of the CMDL score. The sufficient statistics for hypothesis refinement in AVT-NBL can be written as $\mathcal{S}_L(D, h(\Gamma^i) \rightarrow h(\Gamma^{i+1}))$, which can be quantified by the difference between $\mathcal{CMDL}_{\Gamma^{i+1}}$ and $\mathcal{CMDL}_{\Gamma^i}$:

$$s_L(D, h(\Gamma^i) \rightarrow h(\Gamma^{i+1})) = \mathcal{CMDL}_{\Gamma^{i+1}} - \mathcal{CMDL}_{\Gamma^i}$$

Therefore, $h(\Gamma^i)$ will be refined to $h(\Gamma^{i+1})$ when an improved CMDL score is obtained by refining the cut from Γ^i to Γ^{i+1}.

$$h(\Gamma^i) \rightarrow h(\Gamma^{i+1}) \text{ if } \mathcal{S}_L(D, h(\Gamma^i) \rightarrow h(\Gamma^{i+1})) > 0$$

As an example, Figure 6.4 illustrates a hypothesis refinement process. We calculate $\mathcal{CMDL}_\Gamma$ and $\mathcal{CMDL}_{\hat{\Gamma}}$ respectively, and compare their difference. If $\mathcal{CMDL}_\Gamma > \mathcal{CMDL}_{\hat{\Gamma}}$, which means $\mathcal{S}_L(D, h(\Gamma) \rightarrow h(\hat{\Gamma})) > 0$, current hypothesis $h(\Gamma)$ is replaced by $h(\hat{\Gamma})$.

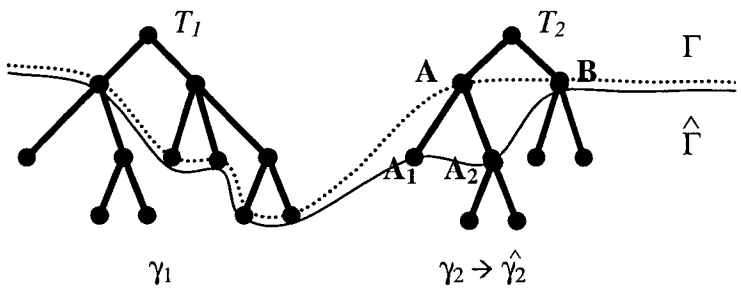


Figure 6.4 A demonstrative hypothesis refinement process

Next, we consider applying AVT-NBL to handle distributed data that is either horizontally fragmented or vertically fragmented. We will show how to decompose queries from AVT-NBL, including the calculations of class conditional frequency counts for all attributes, and the calculations of conditional minimum description length scores with respect to the current cut being explored. In addition, we show how statistics extracted from distributed data sources are combined into a final answer to the original query.

6.2.3.1 Naïve Bayes Classifier from Horizontally Fragmented Distributed Data

First, we consider the case when data are horizontally fragmented. When data are horizontally fragmented, each data source contains a subset of examples. In this setting, $|D| = |D_1| + |D_2| + \ldots + |D_K|$, and each data source has the same ordered set of attributes $\mathbf{A} = \{A_1, A_2, ..., A_N\}$. Each data source D_i has its own attribute value taxonomy $\mathbf{\Lambda}_i$ which can be different from learner AVT $\mathbf{\Lambda}_L$. Presumably we have properly defined all AVT-mappings $\{M(\mathbf{\Lambda}_i, \mathbf{\Lambda}_L)|i = 1, \cdots, K\}$.

The initial query Q on a global class conditional frequency counts (CCFC) with respect to $\mathbf{\Lambda}_L$ is decomposed into sub-queries $q_1, q_2, \ldots, q_K$ with respect to corresponding data sources and their AVTs. Each data source contributes partial information to a global CCFC, and the frequency counts from these K data sources are aggregated towards computing the global frequency counts to reflect real distributions of the whole data D. Specifically, we denote $\sigma_i^{(k)}(v|c_j)$ to be the frequency count of value v of attribute A_i given class label c_j in the training set D_k. By applying $M(\mathbf{\Lambda}_k, \mathbf{\Lambda}_L)$, we obtain its contribution to the correspondent $v_{|M(\mathbf{\Lambda}_k, \mathbf{\Lambda}_L)}$ in $\mathbf{\Lambda}_L$. Moreover, we denote $\mathcal{CCFC}_i^{(k)}$ to be the class conditional frequency counts (also organized as a tree of counts) regarding $\mathcal{T}_i$ of $\mathbf{\Lambda}_L$ contributed by D_k. The query composition procedure can be simply done by summing up all these counts contributed by all K data sources:

for (each data source D_k)
for (each attribute A_i and each value v in $\mathcal{T}_i$)
Compute each $\sigma_i^{(k)}(v_{\|M(\mathbf{\Lambda}_k, \mathbf{\Lambda}_L)}\|c_j)$ in $\mathcal{CCFC}_i^{(k)}$
Composing operation:
for (each attribute A_i)
for (each value v in $\mathcal{T}_i$)
$\mathcal{CCFC}_i^{(L)} = \sum_{k=1}^{K} \mathcal{CCFC}_i^{(k)}$

For calculating each CMDL score with respect to a learner's cut Γ, we only need to decompose the second term on conditional log-likelihood (CLL), the first term regarding the

complexity of the classifier will be calculated directly by the learner.

$$\mathcal{CMDL}_{(\Gamma|D)} = \left(\frac{\log|D|}{2}\right) size(h(\Gamma)) - |D|\mathcal{CLL}_{(\Gamma|D)}$$

We decompose the CLL calculations from horizontally distributed data. Because each data source consists of a subset of the whole training data, the CLL calculation for each data source is relatively independent. The global CLL is the sum of each local CLL score:

for (each data source D_k)
Compute $\mathcal{CLL}_{(\Gamma\|D_k)}$:
$\mathcal{CLL}_{(\Gamma\|D_k)} = \sum_{p=1}^{\|D_k\|} \log\left(\frac{P(c_{X_p})\prod_i P_h(v_{ip}\|c_{X_p})}{\sum_{j=1}^{\|C\|} P(c_j)\prod_i P_h(v_{ip}\|c_j)}\right)$
Composing operation:
$\mathcal{CLL}_{(\Gamma\|D)} = \sum_{k=1}^{K} \mathcal{CLL}_{(\Gamma\|D_k)}$

The class conditional probabilities for attribute values we used in the calculation of CLL score in each data location come from the current hypothesis $h(\Gamma)$. $h(\Gamma)$ consists of class conditional probabilities for attribute values being normalized by the current cut Γ. Therefore, with each statistical query on current CLL, the learner also registers its current hypothesis to the query answering engine, and the hypothesis with its CPTs is accessible to all data sources. It is the function of each data source to calculate its local CLL score based on its local data and send the value back to the query answering engine before a global CLL score is assembled.

6.2.3.2 Naïve Bayes Classifier from Vertically Fragmented Distributed Data

Now, we consider applying AVT-NBL to learn from vertically fragmented data. When data are vertically fragmented, each data location has only a subset of attributes that describe the complete data set. If we denote $\mathbf{A}^j$ to be the set of attributes associated with data source D_j, then we have $\mathbf{A} = \cup_{j=1}^{K} \mathbf{A}^j$, and $|\mathbf{A}| = \sum_{j=1}^{K} |\mathbf{A}^j|$. Thus, one instance in D that is expressed in a tuple of attribute values with a class label has been broken into K sub-tuples with the same class label. Therefore, in order to avoid any ambiguity and inconsistency, each data source contains the same number of instances which have been indexed in the same order. We define

a join operator $\times$, and we can retrieve the complete data by $D = D_1 \times D_2 \times \ldots \times D_K$.

Since the first query is for obtaining CCFC for all attributes, the calculation of each $\mathcal{CCFC}_i$ is independent of other attributes when given the class. All training samples related to a particular attribute are located at one site, so we can calculate $\mathcal{CCFC}_i$ directly from D_j if $A_i \in \mathbf{A}^j$. Thus, this query is decomposed into individual CCFC according to $\mathbf{A}^1, \mathbf{A}^2, \ldots, \mathbf{A}^K$. A final answer to this query is the union of CCFC for all attributes:

$$\{\mathcal{CCFC}_i | i = 1, \cdots, N\} = \cup_{j=1}^{K} \cup_{A_i \in \mathbf{A}^j} \mathcal{CCFC}_i$$

In decomposing statistical query on CMDL score, we need a precise procedure to decompose CLL score calculation from vertically fragmented data sources. By revisiting the formula of calculating CLL score, we notice that the likelihood term with respect to each instance is broken into K pieces by K distributed data sites. What we can calculate is only a partial value of the likelihood term for each sub-tuple of a certain instance. The real likelihood value for the instance is the product of K such partial values. For example, when we want to calculate the likelihood value for an instance X_p that is represented as a tuple of attribute values $(v_{1p}, v_{2p}, \cdots, v_{Np})$, we decompose it into a product of K partial likelihood values corresponding to K such sub-tuples that are distributed among K data sources. Let $PL_k(X_p|c_{X_p})$ denote the partial likelihood value of X_p with regard to data source D_k.

$$\begin{aligned} P_h(c_{X_p}|v_{1p}, \cdots, v_{Np}) &= \prod_i P_h(v_{ip}|c_{X_p}) \\ = \prod_{k=1}^{K} PL_k(X_p|c_{X_p}) &= \prod_{k=1}^{K} \prod_{A_i \in \mathbf{A}^k} P_h(v_{ip}|c_{X_p}) \end{aligned}$$

where $\prod_{A_i \in \mathbf{A}^k} P_h(v_{ip}|c_{X_p})$ is just the partial likelihood value for instance X_p regarding data source D_k. Because of the normalization term, the calculations we need are the partial likelihood values given different classes in $\mathbf{C}$. Therefore, within one vertically fragmented data source D_k, we calculate a matrix $\mathbf{Y}^k$ of order $|D| \times M$, where $|D|$ is the total number of instance

and M is the number of classes.

$$\mathbf{Y}^k_{|D|\times M} = \begin{pmatrix} Y^k_{11} & Y^k_{12} & \dots & Y^k_{1M} \\ Y^k_{21} & Y^k_{22} & \dots & Y^k_{2M} \\ \vdots & \vdots & & \vdots \\ Y^k_{|D|1} & Y^k_{|D|2} & \dots & Y^k_{|D|M} \end{pmatrix}$$

Each entry Y_{pq} in $\mathbf{Y}^k_{|D|\times M}$ equals to $PL_k(X_p|c_q)$, the partial likelihood value for instance X_p given class label c_q. When we obtain all matrices $\{\mathbf{Y}^k|k = 1..K\}$ from K distributed data sites, we can calculate a final CLL score at the central location (use the composition operator of the query answer engine) by the following procedure:

Let $|D| = |D_1| = |D_2| = \ldots = |D_K|$

for (each data source D_k, $k = 1$ to K)

  for (each instance X_p, $p = 1$ to $|D_k|$)

    for (each class c_q, $q = 1$ to M)

      Compute entry $Y_{pq} = PL_k(X_p|c_q)$ in $\mathbf{Y}^k_{|D|\times M}$

Composing operation:

Compute $\mathcal{CLL}_{(\Gamma|D)} =$

$$\log \prod_{p=1}^{|D|} \left(\frac{P(c_{X_p}) \prod_{k=1}^{K} PL_k(X_p|c_{X_p})}{\sum_{q=1}^{|C|} P(c_q) \prod_{k=1}^{K} PL_k(X_p|c_q)} \right)$$

As mentioned earlier, the learner registers its current hypothesis $h(\Gamma)$ with class conditional probability tables to each vertically fragmented data sources. But each data source only needs a subset of the hypothesis (a subset of corresponding CPTs), because each data source has only a subset of the attributes and needs only to calculate partial likelihood values.

6.3 Theoretical Analysis

In order to evaluate the effectiveness and quality of the AVT-NBL algorithm on distributed and semantically heterogeneous data, we use exactness and communication complexity as two criteria to compare the classifier generated by AVT-NBL from distributed and semantically heterogeneous data with that produced by its centralized counterpart.

Proposition 6.1 (EXACTNESS OF AVT-NBL) *AVT-NBL for learning from distributed (either horizontally fragmented or vertically fragmented) and semantically heterogeneous data is exact with respect to AVT-NBL for learning from the complete data obtained by integrating the data sources from a learner perspective.*

Proof Sketch: The exactness of AVT-NBL follows from the correctness of the procedure used for obtaining the relevant sufficient statistics, including CCFC calculations and CMDL scores, from distributed data. The sufficient statistics obtained from semantically heterogeneous data by a precise procedure to decompose the query from the learner into sub-queries and compose the answers to sub-queries is exactly the same as that obtained by its centralized counterpart from the integrated data set.

Based on our general strategy of decomposing statistical queries, we obtained exactly the same statistical query results for both distributed data and corresponding centralized data no matter how data are fragmented. Because the class conditional frequency counts are additive for horizontally fragmented data, and all the data related to calculate the class conditional frequency counts for an attribute are located in one site for vertically fragmented data, the counts will be the same if we calculate them directly from the corresponding centralized data. Moreover, because the conditional log-likelihood scores are decomposable in partial values, and can be reconstructed according to precise procedures, they will also be the same if we calculate them directly from corresponding centralized data. Secondly, we have an unambiguous specification of AVT-mappings to provide a consistent view of the learner. We also have a build-in mechanism in AVT-NBL to deal with partially specified data. All of these combined can reconcile the semantic heterogeneity among different data sources. Therefore, AVT-NBL is exact in learning from distributed and semantically heterogeneous data.

Besides the exactness of the learning algorithm, the communication cost is another important factor to be considered in learning from distributed data. For the simplicity of analysis, we assume that each data site can ship both the raw data and the extracted sufficient statistics. Under the assumption that both local computation of the sufficient statistics and shipping of the raw data are possible, we are interested in exploring a condition where algorithms for

learning from distributed data is preferable to the algorithms for learning from centralized data.

Recall from previous notions, N is the number of attributes, M is the number of classes, $|D|$ is the total number of training instances, K is the number of distributed data sources. Several extra notions include: $|AVT|$ is the maximum number of attribute values (both primitive and abstract) of an attribute, also known as the size of the AVT; $\widetilde{H}_{max}$ is the maximum number of refinements on partial hypothesis.

Proposition 6.2 (COMMUNICATION COMPLEXITY) *AVT-NBL for learning using sufficient statistics is preferable to AVT-NBL for learning using the centralized data in terms of communication complexity when:*

(1). Data are horizontally fragmented, and $O(|AVT| \cdot N \cdot M \cdot K + K \cdot \widetilde{H}_{max}) < |D| \cdot N$

(2). Data are vertically fragmented, and $O(|AVT| \cdot N \cdot M + K \cdot |D| \cdot M \cdot \widetilde{H}_{max}) < |D| \cdot N$

Proof : In the centralized case, all raw data are shipped to a central location. The total number of messages sent is $(|D_1| + \ldots + |D_k|)(N+1) = |D|(N+1)$. We estimate the communication complexity for the two types of data fragmentation separately:

(1). When data are horizontally fragmented, the total number of messages to answer the query on CCFC is $(|AVT| \cdot N \cdot M \cdot K)$. Recall that subsequent queries are on CMDL scores. For calculating each CMDL score, we ship K CLL scores from each data source. Since the maximum number of refinements is $\widetilde{H}_{max}$, the maximum number of messages to answer those queries on CMDL scores is $(K \cdot \widetilde{H}_{max})$. In order for each data source to access the hypothesis, the maximum number of messages that need to be sent to K data sources is $(|AVT| \cdot N \cdot M \cdot K)$. Thus, the total number of messages is $(2|AVT| \cdot N \cdot M \cdot K) + (K \cdot \widetilde{H}_{max})$. The condition to favor learning using sufficient statistics is that the total number of messages is less than $|D|(N+1)$, which implies $O(|AVT| \cdot N \cdot M \cdot K + K \cdot \widetilde{H}_{max}) < |D| \cdot N$.

(2). When data are vertically fragmented, the total number of messages to answer the query on CCFC is $(|AVT| \cdot N \cdot M)$. For calculating each CMDL score in subsequent queries, we need to calculate matrices $\{\mathbf{Y}^k | k = 1..K\}$ from K distributed data sources. The dimension of each $\mathbf{Y}^k$ is $|D| \times M$, so the number of messages needed is $(K \cdot |D| \cdot M)$. Similarly, the maximum

number of messages to answer queries on CMDL scores will be $(K \cdot |D| \cdot M \cdot \widetilde{H}_{max})$. For sending partial hypothesis to K data sources, maximum number of messages is $(|AVT| \cdot N \cdot M)$. Thus, the total number of messages is $(2|AVT| \cdot N \cdot M + K \cdot |D| \cdot M \cdot \widetilde{H}_{max})$. The condition to favor learning using sufficient statistics is $O(|AVT| \cdot N \cdot M + K \cdot |D| \cdot M \cdot \widetilde{H}_{max}) < |D| \cdot N$.

6.4 Experimental Evaluation

Previous experimental results of AVT-NBL on benchmark data sets showed that AVT-NBL is able to generate classifiers that are substantially more compact and more accurate than those produced by NBL on a broad range of data sets with different percentages of partially specified values, and AVT-NBL is more efficient in its use of training data. Please refer to chapter 4 for a detailed description of the previous results.

In this section we provide the *proof-of-concept* of our approach and show that AVT-NBL in learning from distributed and semantically heterogeneous data are *exact* to their batch (or centralized) counterparts. We applied AVT-NBL to distributed, semantically heterogeneous data fragments from several different domains and compared the results with the results obtained by applying AVT-NBL to the data set obtained by appropriately integrating the distributed fragments according to a user perspective. We also compare the performance of AVT-NBL relative to that of the standard NBL on distributed and semantically heterogeneous data.

6.4.1 Experimental Setup

Even though partially specified data and hierarchical AVT are common in many application domains, to the best of our knowledge, there are no widely used benchmark data sets for evaluation of classifiers from semantically heterogeneous distributed data sources. As a result, we managed to design such benchmark data based on three data sets (with only nominal attributes) selected from the UC Irvine Machine Learning Repository. For all three of them (i.e., *Car, Nursery,* and *Mushroom*), the AVT's were supplied by domain experts.

To simulate horizontally fragmented data sets, we arbitrarily split the original training data set into two subsets, D_1^h and D_2^h of approximately the same size, and in order to simulate

vertically fragmented data sets, we randomly split the original set of attributes (from the training data set) into two attribute subsets of roughly the same size. The data corresponding to the first attribute subset comprises the first fragment, D_1^v, whereas the remaining comprises the second fragment, D_2^v. Note that, for the sake of simplicity, we assumed two fragments (horizontally or vertically), although, in general, it is possible to have more than two fragments.

Our first set of experiments compares the performance of AVT-NBL and NBL in the centralized and distributed learning scenario where each data source AVT is the same as that of the learner. Our second set of experiments is similar to the previous settings, however, we assume that the learner AVT is different from data source AVTs. As a consequence, data are specified at different levels of abstraction.

6.4.2 Results and Observations

We achieve exact accuracy in learning from centralized and distributed data. Table 6.2 and Table 6.3 shows the accuracies of the classifiers generated by AVT-NBL and NBL respectively on *Car*, *Mushroom*, and *Nursery* data. We apply AVT-NBL and NBL to each data when they are centralized, horizontally distributed, and vertically distributed. As can be seen from Tables 6.2, we obtain the *exact* same accuracies for AVT-NBL that is applied to centralized, horizontally distributed and vertically distributed data when we have the same data source AVTs as the learner AVT. According to Table 6.3, we also observe the *exact* same accuracy for centralized and distributed data when data source AVTs are different from learner AVT. The same results hold for NBL too.

AVT-NBL performs better than NBL. Tables 6.2 and Table 6.3 show the accuracies obtained for the classifiers generated by AVT-NBL and NBL. As can be seen, the accuracy for AVT-NBL is significantly higher than the traditional NBL when data are distributed and semantically heterogeneous. This difference is more prominent in the case of learning from semantically heterogeneous data when partially specified data are inevitable in this learning scenario. Because NBL treat partially specified attribute values as if they were totally missing, while AVT-NBL has a built-in mechanism to deal with partially specified data directly, and

Table 6.2 Comparison of accuracy and size of classifiers generated by AVT-NBL and NBL on distributed data without heterogeneity.

Data Sets	Centralized Data (for AVT-NBL)		Horizontally Distributed Data (for AVT-NBL)		Vertically Distributed Data (for AVT-NBL)	
	Accuracy	Size	Accuracy	Size	Accuracy	Size
Car	85.59%	48	85.59%	48	85.59%	48
Mushroom	99.85%	124	99.85%	124	99.85%	124
Nursery	90.16%	95	90.16%	95	90.16%	95
Data Sets	**Centralized Data (for NBL)**		**Horizontally Distributed Data (for NBL)**		**Vertically Distributed Data (for NBL)**	
Car	85.06%	76	85.06%	76	85.06%	76
Mushroom	95.56%	240	95.56%	240	95.56%	240
Nursery	90.11%	125	90.11%	125	90.11%	125

perform regularization at high levels of abstraction, AVT-NBL shows much better performance than AVT-NBL when applied to semantically heterogeneous data.

Table 6.3 Comparison of accuracy and size of classifiers generated by AVT-NBL and NBL on semantically heterogeneous data.

Data Sets	Centralized Data (for AVT-NBL)		Horizontally Distributed Data (for AVT-NBL)		Vertically Distributed Data (for AVT-NBL)	
	Accuracy	Size	Accuracy	Size	Accuracy	Size
Car	81.94%	48	81.94%	48	81.94%	48
Mushroom	99.07%	124	99.07%	124	99.07%	124
Nursery	86.87%	95	86.87%	95	86.87%	95
Data Sets	**Centralized Data (for NBL)**		**Horizontally Distributed Data (for NBL)**		**Vertically Distributed Data (for NBL)**	
Car	63.76%	76	63.76%	76	63.76%	76
Mushroom	92.83%	240	92.83%	240	92.83%	240
Nursery	64.77%	125	64.77%	125	64.77%	125

AVT-NBL generates compact classifiers than traditional NBL. The column *Size* in Tables 6.2 and Table 6.3 refer to the size of the classifier produced by each learning algorithm. They refer to the total number of class conditional probabilities needed to specify the clas-

sifier generated. The results show that AVT-NBL is effective in exploiting the information supplied by the AVT to generate compact classifiers. Thus, AVT-based learning algorithms offer an approach to compressing class conditional probability distributions even when data are distributed and semantically heterogeneous. This is consistent with our observations in AVT-NBL applied to centralized data.

6.5 Summary and Discussion

Learning from semantically heterogeneous data is a key problem in many application domains, including bioinformatics, geoinformatics, semantic web, etc. There is an increasing need for such algorithms for learning pattern classifiers from such data sources and extracting interesting rules from a learner's perspective.

In this chapter, we precisely formulate the problem of learning classifiers from distributed, semantically heterogeneous data sources (i.e., ontology-extended data sources) viewed from a learner's perspective. We extend a previous approach for learning Naïve Bayes classifier from data with associated attribute value taxonomies (semantically homogeneous) and partially specified data [Zhang and Honavar (2004a)] to an approach for learning compact and accurate Naïve Bayes classifier from distributed and heterogeneous data sources. We present a principled way to reduce the problem of learning from semantically heterogeneous data to the problem of learning from distributed partially specified data by reconciling semantic heterogeneity using AVT-mappings, and describe a sufficient statistics based solution to distributed data that is either horizontally fragmented or vertically fragmented. To apply AVT-NBL to horizontally fragmented or vertically fragmented distributed data, we decompose each statistical query into sub-queries in a principled way according to the need of the AVT-NBL algorithm, and combine the answers to the sub-queries into an answer to the original query.

A brief theoretical analysis of the resulting algorithm shows that AVT-NBL for learning from distributed (either horizontally fragmented or vertically fragmented) and semantically heterogeneous data is *exact* with respect to AVT-NBL for learning from the complete data obtained by integrating the data sources from a learner perspective.

Our experiment results have verified our theoretical analysis on the exactness of AVT-NBL. We obtained the same accuracies in the case of centralized, horizontal and vertical data distributions. Our experiment results also showed that AVT-NBL is able to generate classifiers that are substantially more compact and accurate than those produced by NBL on heterogeneous distributed data. Because AVT-NBL has a built-in mechanism to deal with partially specified data that are unavoidable in integrating information from semantically heterogeneous data, AVT-NBL outperforms standard NBL, which can not handle partially specified data properly.

In this chapter, we have shown our approach to learning the Naïve Bayes classifier from ontology-extended data sources. Based on our discussion of AVT-based classifier learners in a previous chapter, our solution to this learning problem can be generalized to transform a large class of algorithms for learning from data into algorithms for learning from distributed, semantically heterogeneous data, including learning decision trees, neural networks, support vector machines, and among others. Problems of interests also include learning classifiers from semantically heterogenous relational data sources, and applying the resulting algorithms to problems in scientific domains (e.g., bioinformatics).

CHAPTER 7. Summary, Conclusions and Future Research

7.1 Summary

Current advances in machine learning have offered powerful approaches to exploring complex, a-priori unknown relationships or discovering hypotheses that describe potentially interesting regularities from data. Data-driven knowledge discovery in practice, occurs within a *context*, or under certain *ontological commitments* on the part of the learner. Making ontological commitments (that are typically implicit in a data set) *explicit* enables users to explore data from different points of view, and at different levels of abstraction. Each point of view corresponds to a set of ontological (and representational) commitments regarding the domain of interest. In scientific discovery, there is no single perspective that can serve all purposes, and it is always helpful to analyze data in different contexts and from alternative representations. Hence, there is a pressing need for ontology-aware learning algorithms to facilitate the exploration of data from multiple points of view.

In this dissertation, we have precisely formulated the problem of learning pattern classifiers from attribute value taxonomies and data (which can be partially specified), and designed AVT-guided learning algorithms which extend standard learning algorithms in principled ways so as to exploit the information provided by AVT. We have designed and implemented AVT-NBL and AVT-DTL for learning AVT-guided Naïve Bayes and Decision Tree classifiers, respectively. An AVT-guided learning algorithm has a bias in favor of the level of abstraction based on more abstract attribute values (i.e., those that appear closer to the roots of the corresponding AVTs) that are sufficiently informative for classifying the training set. Our AVT-guided learning algorithms adopt a general learning framework that takes into account the tradeoff between the complexity and the accuracy of the predictive models. This tradeoff enable us to learn the

classifier that is both compact and accurate.

Our experimental results presented in this dissertation have shown that:

- AVT-guided learning algorithms (i.e., AVT-DTL and AVT-NBL) are able to learn robust, high accuracy classifiers from data sets consisting of a relatively high percentage of partially specified instances.
- AVT-guided learning algorithms yield substantially more compact, yet high accuracy classifiers than standard learning algorithms when applied to data sets with fully specified instances, as well as data sets with a relatively high percentage of missing attribute values.
- AVT-guided learning algorithms achieve a better trade-off between accuracy and complexity of the resulting classifiers than standard learning algorithms applied to propositionalized data set.
- AVT-guided learning algorithms are more efficient in their use of training data. AVT-guided learning algorithms can produce classifiers that outperform those produced by standard learning algorithms using less training samples.

We have provided a general framework for learning classifiers from attribute value taxonomies and data. We have illustrated the application of this framework in the case of AVT-based variants of decision tree and Naïve Bayes classifiers. However, this framework can be used to derive AVT-based variants of other learning algorithms.

We have extended our previous approach to learning compact and accurate classifiers from partially specified semantically heterogeneous data sources. Our approach to AVT-guided learning from partially specified semantically heterogeneous data relies on our general strategy for transforming algorithms for learning from data into algorithms for learning from distributed, semantically heterogeneous data. This strategy is based on the decomposition of the learning task into an information extraction component (when *sufficient statistics* needed for learning are gathered) and a hypothesis generation component (that uses the *sufficient statistics* to generate or refine a current hypothesis).

We present a principled way to reduce the problem of learning from semantically heterogeneous data to the problem of learning from distributed partially specified data by reconciling semantic heterogeneity using AVT-mappings, and describe a sufficient statistics based solution. The resulting algorithm is exact relative to its centralized counterpart, and our experimental results on synthesized distributed and semantically heterogeneous data verified our theoretical analysis on the exactness of the algorithm. We illustrate our approach to using this strategy to design AVT-guided algorithms for learning classifiers from semantically heterogeneous data using the Naïve Bayes classifier as an example. However, the proposed approach can be extended to a broad range of other machine learning algorithms.

7.2 Contributions

The main contributions of this dissertation include:

1. **AVT-based Decision Tree Learner (AVT-DTL) for learning Decision Tree classifier from attribute value taxonomies and data** [Zhang and Honavar (2003)].

   AVT-DTL is a generalized version of standard decision tree learning algorithm for learning classifiers from attribute value taxonomies and data. AVT-DTL implements a top-down AVT-guided search in decision tree hypothesis space, and it has a bias in favor of splits based on more abstract attribute values that are sufficiently informative for classifying the training set. Our experimental results have shown that AVT-DTL produces compact and easy-to-comprehend decision tree classifiers, and yields more accurate decision tree classifiers. AVT-DTL also yields significant lower error rates than C4.5 on data sets with substantially large percentages of partially missing attribute values.

2. **AVT-based Naïve Bayes Learner (AVT-NBL) for learning Naïve Bayes classifier from attribute value taxonomies and data** [Zhang and Honavar (2004a)].

   AVT-NBL is an extension of the standard Naïve Bayes learning algorithm that effectively exploits user-supplied AVTs to construct compact and accurate Naïve Bayes classifier from partially specified data. Our experimental results have shown that AVT-NBL is

able to generate classifiers that are substantially more compact and more accurate than those produced by NBL on a broad range of data sets with different percentages of partially specified values. We have shown that AVT-NBL is more efficient in its use of training data: AVT-NBL produces classifiers that outperform those produced by NBL using substantially fewer training examples.

3. **A general framework for design of algorithms for learning classifiers from attribute value taxonomies and data** [Zhang et al. (2005a)].

   We have proposed a general framework for the design of algorithms for learning classifiers from attribute value taxonomies and data (i.e., ontology-aware algorithms). We identify three elements in learning classifiers from AVT-extended data sources: (1) A procedure for identifying estimated sufficient statistics on AVTs from data; (2) A procedure for building and refining hypothesis; (3) A performance criteria for making the tradeoff between complexity and accuracy of the generated classifiers. We illustrate the instantiation of this framework in the case of AVT-DTL and AVT-NBL.

4. **A general framework for learning concise and accurate classifiers from semantically heterogeneous data.**

   We extend our previous approach for learning Naïve Bayes classifier from semantically homogeneous data with associated attribute value taxonomies and partially specified data to an approach for learning compact and accurate Naïve Bayes classifier from distributed and heterogeneous data sources. We present a principled way to reduce the problem of learning from semantically heterogeneous data to the problem of learning from distributed partially specified data by reconciling semantic heterogeneity using AVT-mappings, and describe a sufficient statistics based solution. We prove that the resulting algorithm is exact relative to its centralized counterpart.

7.3 Future Work

Some promising directions for future work include:

1. Development of AVT-based variants of other machine learning algorithms for construction of classifiers from partially specified data, and from distributed, semantically heterogeneous data sources. Specifically, it would be interesting to design AVT-based variants of algorithms for constructing Bag-of-words classifiers, Bayesian Networks, Nonlinear Regression Classifiers, and Hyperplane classifiers (Perceptron, Winnow Perceptron, and Support Vector Machines).

2. Extensions that incorporate richer classes of AVT. Our work has so far focused on tree-structured taxonomies defined over nominal attribute values. It would be interesting to extend this work in several directions motivated by the natural characteristics of data: (a) Hierarchies of intervals to handle numerical attribute values; (b) Ordered generalization hierarchies where there is an ordering relation among nodes at a given level of a hierarchy (e.g., hierarchies over education levels); (c) Tangled Hierarchies that are represented by directed acyclic graphs (DAG) and Incomplete Hierarchies which can be represented by a forest of trees or DAGs.

3. Extensions that incorporate class taxonomies (CT). It would be interesting to explore approaches that exploit the hierarchical structure over class labels directly in constructing classifiers. It would also be interesting to explore several possibilities for combining approaches to exploiting CT with approaches to exploiting AVT to design algorithms that make the optimal use of CT and AVT to learn robust, compact and easy-to-interpret classifiers from partially specified data. At this stage, we have some preliminary work on learning classifiers using hierarchically structured class taxonomies [Wu et al. (2005)]. Some ongoing research in this direction includes development of algorithms to incorporate techniques for exploiting CT (class taxonomies) to handle partially specified class labels, and development of more sophisticated metrics for evaluation of structured label classifiers.

4. Further experimental evaluation of AVT-NBL, AVT-DTL, and related learning algorithms on a broad range of data sets in scientific knowledge discovery applications, in-

cluding: (a) Census data from official data libraries [1]; (b) Data sets for macromolecular sequence-structure-function relationships discovery, including Gene Ontology Consortium [2] and MIPS [3]; (c) Data sets of system and network logs for intrusion detection.

5. Application of the general framework of learning classifiers from partially specified semantically heterogeneous data to multi-relational databases from ontology-extended data sources (with possible partially specified data from multi-relational databases). Based on the previous work of learning classifiers from relational tables, it is of interest to explore approaches for developing sophisticated approaches to estimate statistics needed by learning algorithms to build classifiers from ontology-extended multi-relational data sources.

6. Application of the resulting algorithms and software to collaborative discovery problems that arise in areas such as computational biology (e.g., discovery of relationships between macromolecular sequence, structure, expression, interaction, function, and evolution); discovery of genetic regulatory networks from multiple sources of data (e.g., gene expression, protein localization, protein-protein interaction).

[1] http://www.thedataweb.org/
[2] http://www.geneontology.org/
[3] http://mips.gsf.de/

BIBLIOGRAPHY

Akaike, H. (1974). A new look at statistical model identification. *IEEE Trans. on Automatic Control*, AU-19:716–722.

Almuallim, H., Akiba, Y., and Kaneda, S. (1995). On handling tree-structured attributes. In *Proceedings of the Twelfth International Conference on Machine Learning.*

Almuallim, H., Akiba, Y., and Kaneda, S. (1996). An efficient algorithm for finding optimal gain-ratio multiple-split tests on hierarchical attributes in decision tree learning. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence and Eighth Innovative Applications of Artificial Intelligence Conference.*

Alpaydin, E. (2004). *Introduction to Machine Learning.* MIT Press, Cambridge, MA.

Aronis, J. and Provost, F. (1996). Exploiting background knowledge in automated discovery. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pages 355–358, AAAI Press.

Aronis, J. and Provost, F. (1997). Increasing the efficiency of inductive learning with breadth-first marker propagation. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*, pages 119–122, AAAI Press.

Ashburner, M. and et al. (2000). Gene ontology: tool for the unification of biology. the gene ontology consortium. *Nature Genetics*, 25:25–29.

Bergadano, F. and Giordana, A. (1990). Guiding induction with domain theories. *Machine Learning - An Artificial Intelligence Approach*, 3:474–492.

Berners-Lee, T., Hendler, J., and Lassila, O. (2001a). The semantic web. *Scientific American.*

Berners-Lee, T., J., H., and Lassila, O. (2001b). The semantic web. *Scientific American*, pages 35–43.

Bhatnagar, R. and Srinivasan, S. (1997). Pattern discovery in distributed databases. In *Proceedings of the Fourteenth AAAI*, pages 503–508, Providence, RI. AAAI Press/The MIT Press.

Bhattacharya, I. and Getoor, L. (2004). Deduplication and group detection using links. In *KDD Workshop on Link Analysis and Group Detection, Aug. 2004, Seattle.*

Blockeel, H., Bruynooghe, M., Dzeroski, S., Ramon, J., and Struyf, J. (2002). Hierarchical multi-classification. In *De Raedt, L. and Dzeroski, S. and Wrobel, S. (Ed). Proceedings of the Multi-relational data mining workshop (MRDM 2002).*

Bonatti, P., Deng, Y., and Subrahmanian, V. (2003). An ontology-extended relational algebra. In *Proceedings of the IEEE Conference on INformation Integration and Reuse*, pages 192–199. IEEE Press.

Bonnissone, P. and Tong, R. (1985). Editorial: reasoning with uncertainty in expert systems. *International Journal of Man Machine Studies*, 22:241–250.

Breiman, L. (1998). Bias-variance, regularization, instability and stabilization. *Neural Networks and Machine Learning.*

Breiman, L., Friedman, J., Olshen, R., and Stone, C. (1984). *Classification and Regression Trees.* Wadsworth & Brooks, Monterey, CA.

Buja, A. and Lee, Y.-S. (2001). Data mining criteria for tree-based regression and classification. In *KDD*, pages 27–36.

Caragea, D., Pathak, J., and Honavar, V. (2004a). Learning classifiers from semantically heterogeneous data. In *Proceedings of the 3rd International Conference on Ontologies, Databases, and Applications of Semantics for Large Scale Information Systems, ODBASE-2004.*

Caragea, D., Silvescu, A., and Honavar, V. (2000). Agents that learn from distributed and dynamic data sources. In *Proceedings of the Workshop on Learning Agents, Agents 2000/ECML 2000*, pages 53–61.

Caragea, D., Silvescu, A., and Honavar, V. (2004b). A framework for learning from distributed data using sufficient statistics and its application to learning decision trees. In *International Journal of Hybrid Intelligent Systems. Vol. 1, 2004.*

Caragea, D., Zhang, J., Bao, J., Pathak, J., and Honavar, V. (2005a). Algorithms and software for collaborative discovery from autonomous, semantically heterogeneous, distributed information sources. In *Proceedings of the 16th International Conference on Algorithmic Learning Theory. Lecture Notes in Computer Science. Singapore*, Vol. 3734, pages 13–44, Berlin: Springer-Verlag.

Caragea, D., Zhang, J., Pathak, J., and Honavar, V. (2005b). Learning concise and accurate classifiers from semantically heterogeneous data distributed data. *Under Review.*

Casella, G. and Berger, R. (2001). *Statistical Inference.* Duxbury Press, Belmont, CA.

Chen, A., Chiu, J., and Tseng, F. (1996). Evaluating aggregate operations over imprecise data. *IEEE Tans. on Knowledge and Data Engineering*, 8(28):273–284.

Chen, J., Shapcott, M., McClean, S., and Adamson, K. (2002). Learning with concept hierarchies in probabilistic relational data mining. In *Proceedings of the Third International Conference on Web Age Information Management.*

Chen, R. and Sivakumar, K. (2002). A new algorithm for learning parameters of a bayesian network from distributed data. In *Proceedings of The IEEE International Conference on Data Mining, 2002.*

Chen, R., Sivakumar, K., and Kargupta, H. (2003). Learning bayesian network stucture from distributed data. In *Proceedings of Third SIAM International Conference on Data Mining.*

Cherkassky, C. and Mulier, F. (1998). *Learning From Data.* John Wiley and Sons, New York.

Cho, V. and Wuthrich, B. (2002). Distributed mining of classification rules. *Knowledge and Information Systems*, 4(1).

Ciampi, A., Chang, C., Hogg, S., and McKinney, S. (1987). Recursive partition: a versatile method for exploratory data analysis in biostatistics. *Biostatistics*, pages 23–50.

Clare, A. and King, R. (2001). Knowledge discovery in multi-label phenotype data. In *Proceedings of the Fifth European Conference on Principles of Data Mining and Knowledge Discovery. Lecture Notes in Computer Science*, Vol. 2168, pages 42–53, Springer.

Clark, L. and Pregibon, D. (1992). Tree-based models. *Statistical Models in S*, pages 377–419.

Cohen, P. (1995). *Empirical Methods for Artificial Intelligence.* MIT Press, Cambridge, MA.

Cohen, W. (1996a). Learning trees and rules with set-valued features. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 709–716.

Cohen, W. (1996b). Learning trees and rules with set-valued features. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence, AAAI/MIT Press.*

Davidson, S., Crabtree, J., Brunk, B., Schug, J., Tannen, V., Overton, G., and Stoeckert, C. (2001). Experiments in integrated access to genomic data sources. *IBM Journal*, 40(2).

DeMichiel, L. (1989). Resolving database incompatibility: An approach to performing relational operations over mismatched domains. *IEEE Tans. on Knowledge and Data Engineering*, 1(4):485–493.

Dempster, A., Laird, N., and Rubin, D. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38.

desJardins, M., Getoor, L., and Koller, D. (2000). Using feature hierarchies in bayesian network learning. In *Proceedings of Symposium on Abstraction, Reformulation, and Approximation 2000. Lecture Notes in Artificial Intelligence*, Vol. 1864, pages 260–270, Springer.

Dhar, V. and Tuzhilin, A. (1993). Abstract-driven pattern discovery in databases. *IEEE Tans. on Knowledge and Data Engineering*, 5(6):926–938.

Dietterich, T. (2003). Machine learning. *Nature Encyclopedia of Cognitive Science.*

Diettterich, T. (2003). Machine learning. *Nature Encyclopedia of Cognitive Science.*

Domingos, P. (1997). Knowledge acquisition from examples via multiple models. In *Proceedings of the Fourteenth International Conference on Machine Learning*, pages 98–106, Nashville, TN. Morgan Kaufmann.

Domingos, P. and Pazzani, M. (1997). On the optimality of the simple bayesian classifier under zero-one loss. *Machine Learning*, 29(2-3):103–130.

Duda, R., Hart, P., and Stork, D. (2001). *Pattern classification.* New York: Wiley.

Eckman, B. (2003). A practitioner's guide to data management and data integration in bioinformatics. *Bioinformatics*, pages 3–74.

Fan, W., Stolfo, S., and Zhang, J. (1999). The application of adaboost for distributed, scalable and on-line learning. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.*

Ferri-Ramfrez, C., Hernandez-Orallo, J., and Ramirez-Quintana, M. (2001). Learning mdl-guided decision trees for constructor-based languages. In *Proceedings of 11th International Conference on Inductive Logic Programming.*

Forman, G. and Zhang, B. (2000). Distributed data clustering can be efficient and exact. In *Proceedings of the SIGKDD Explorations*, pages 34–38.

Friedman, J. (1997a). On bias, variance, 0/1loss, and the curse-of-dimensionality. *Data Mining and Knowledge Discovery*, 1:55–77.

Friedman, J., Kohavi, R., and Yun, Y. (1996). Lazy decision trees. In *Proceedings of The 13th National Conference on Artificial Intelligence.*

Friedman, N. (1997b). Learning belief networks in the presence of missing values and hidden variables. In *Proceedings of the 14th Conference on Machine Learning.*

Friedman, N., Geiger, D., and Goldszmidt, M. (1997). Bayesian network classifiers. *Machine Learning*, 29.

Gorodetski, V., Skormin, V., Popyack, L., and Karsayev, O. (2000). Distributed learning in a data fusion system. In *Proceedings of the Conference of the World Computer Congress (WCC-2000).*

Han, J. and Fu, Y. (1996). Attribute-oriented induction in data mining. *Advances in Knowledge Discovery and Data Mining*, pages 399–421.

Han, J. and Kamber, M. (2001). *Data Mining: Concepts and Techniques.* Morgan Kaufmann.

Haussler, D. (1988). Quantifying inductive bias: Ai learning algorithms and valiant's learning framework. *Artificial Intelligence*, 36:177–221.

Heckerman, D. (1999). *A Tutorial on Learning with Bayesian Networks.* In Learning in Graphical Models, M. Jordan, ed.. MIT Press, Cambridge, MA.

Hendler, J., Stoffel, K., and Taylor, M. (1996). Advances in high performance knowledge representation. In *University of Maryland Institute for Advanced Computer Studies Dept. of Computer Science, Univ. of Maryland, July 1996. CS-TR-3672 (Also cross-referenced as UMIACS-TR-96-56).*

Hunt, E., Marin, J., and Stone, P. (1966). *Experiments in Induction.* Academic Press, New York.

Jensen, V. and Soparkar, N. (2000). Frequent itemset counting across multiple tables. In *Proceedings of Pacific-Asia Conference on Knowledge Discovery and Data Mining.*

Joo, J., Zhang, J., Yang, J., and Honavar, V. (2004). Generating avts using ga for learning decision tree classifiers with missing data. In *Proceedings of the 7th International Conference on Discovery Science (DS'04).*

Kang, D.-K., Silvescu, A., Zhang, J., and Honavar, V. (2004). Generation of attribute value taxonomies from data for data-driven construction of accurate and compact classifiers. In *Proceedings of the Fourth IEEE International Conference on Data Mining.*

Karalic, A. and Pirnat, V. (1991). Significance level based classification with multiple trees. *Informatica*, 15(5).

Kargupta, H. (2000). Distributed data mining: Towards the next generation of knowledge discovery systems. In *Second Workshop on Mining Scientific Datasets.*

Kargupta, H., Huang, W., Sivakumar, K., and Johnson, E. (2001). Distributed clustering using collective principal component analysis. *Knowledge and Information Systems*, 3(4):422–448.

Kargupta, H., Park, B., Hershberger, D., and Johnson, E. (1999). Collective data mining: A new perspective toward distributed data mining. In Kargupta, H. and Chan, P., editors, *Advances in Distributed and Parallel Knowledge Discovery.* MIT/AAAI Press.

Kargupta, H. and Sivakumar, K. (2004). Existential pleasures of distributed data mining. *Data Mining: Next Generation Challenges and Future Directions*, pages 1–25.

Kohavi, R., Becker, B., and Sommerfield, D. (1997). Improving simple bayes. In *Tech. Report, Data Mining and Visualization Group, Silicon Graphics Inc.*

Kohavi, R. and Provost, P. (2001). Applications of data mining to electronic commerce. *Data Mining and Knowledge Discovery*, 5(1/2):5–10.

Koller, D. and Sahami, M. (1997). Hierarchically classifying documents using very few words. In *Proceedings of the Fourteenth International Conference on Machine Learning*, pages 170–178, Morgan Kaufmann.

Kolluri, V. and Metzler, D. (1999). Knowledge guided rule learning. In *Proceedings of Annual Meeting of American Society for Information Science, November, 1999.*

Kolmogorov, A. (1965). Three approaches to the quantitative definition of information. *Problems of Information Transmission.*

Kudoh, Y., Haraguchi, M., and Okubo, Y. (2003). Data abstractions for decision tree induction. *Theoretical Computer Science*, 292:387–416.

Lang, K. (1995). Newsweeder: Learning to filter netnews. In *Proceedings of the Twelfth International Conference on Machine Learning.*

Langley, P., Iba, W., and Thompson, K. (1992). An analysis of bayesian classifiers. In *Proceedings of the Tenth National Conference on Artificial Intelligence.*

Lenat, D. (1995). Cyc: A large-scale investment in knowledge infrastructure. In *Communications of the ACM*, pages. 33-38, vol. 38, no. 11, Nov, 1995.

Levy, A. Y. (2000). Logic-based techniques in data integration. In *Logic-based artificial intelligence*, pages 575–595. Kluwer Academic Publishers.

Little, R. and Rubin, D. (1987). *Statistical analysis with missing data.* New York: Wiley and Sons.

Liu, W. and White, A. (1997). Techniques for dealing with missing values in classification. In *IDA 97, Vol. 1280 of Lecture Notes.*

Maluf, D. and Wiederhold, G. (1997). Abstraction of representation in interoperation. *Lecture Notes in AI*, 1315.

Mansour, Y. (1994). Learning boolean functions via the fourier transform. In *Theoretical Advances in Neural Computation and Learning.* Kluwer Academic Publishers.

McCallum, A., Rosenfeld, R., Mitchell, T., and Ng, A. (1998). Improving text classification by shrinkage in a hierarchy of classes. In *Proceedings of the Fifteenth International Conference on Machine Learning.*

McClean, S., Páircéir, R., Scotney, B., and Greer, K. (2002). A negotiation agent for distributed heterogeneous statistical databases. *SSDBM 2002*, pages 207–216.

McClean, S., Scotney, B., and Greer, K. (2003). A scalable approach to integrating heterogeneous aggregate views of distributed databases. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, pages 232–235.

McClean, S., Scotney, B., and Shapcott, M. (2001). Aggregation of imprecise and uncertain information in databases. *IEEE Tans. on Knowledge and Data Engineering*, 13(6):902–912.

Mehta, M., Rissanen, J., and Agrawal, R. (1995). Mdl-based decision tree pruning. In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining (KDD-95).*

Miller, G. (1995). Wordnet: A lexical database for english. In *Communications of the ACM, vol. 38, no. 11, Nov, 1995.*

Mitchell, T. (1997). *Machine Learning.* MCB McGraw-Hill.

Núñez, M. (1991). The use of background knowledge in decision tree induction. *Machine Learning*, 6:231–250.

Parsons, S. (1996). Current approaches to handling imperfect information in data and knowledge bases. *IEEE Tans. on Knowledge and Data Engineering*, 8(3):353–372.

Pazzani, M. and Kibler, D. (1992). The role of prior knowledge in inductive learning. *Machine Learning*, 9:54–97.

Pazzani, M., Mani, S., and Shankle, W. (1997). Beyond concise and colorful: Learning intelligible rules. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining.*

Pereira, F., Tishby, N., and Lee, L. (1993). Distributional clustering of english words. In *Proceedings of the Thirty-first Annual Meeting of the Association for Computational Linguistics*, pages 183–190.

Prodromidis, A., Chan, P., and Stolfo, S. (2000). Meta-learning in distributed data mining systems: Issues and approaches. In Kargupta, H. and Chan, P., editors, *Advances of Distributed Data Mining*. AAAI Press.

Pyle, D. (1999). *Data Preparation for Data Mining*. Morgan Kaufmann.

Quinlan, R. (1979). Discovering rules by induction from large collections of examples. pages 168–201.

Quinlan, R. (1986). Induction of decision trees. *Machine Learning*, 1:81–106.

Quinlan, R. (1987). Simplifying decision trees. *International Journal of Man-Machine Studies*, 27(3):221–234.

Quinlan, R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA.

Quinlan, R. (1995). The minimum description length principle and categorical theories. In *Proceedings of 7th International Conference on Machine Learning*.

Quinlan, R. and Rivest, R. (1989). Inferring decision trees using the minimum description length principle. *Information and Computation*, 80:227–248.

Reinoso-Castillo, J., Silvescu, A., Caragea, D., Pathak, J., and Honavar, V. (2003). Information extraction and integration from heterogeneous, distributed, autonomous information sources: A federated, query-centric approach. In *IEEE International Conference on Information Integration and Reuse*. In press.

Rissanen, J. (1978). Modeling by shortest data description. *Automatica*, 14.

Sahami, M., Dumais, S., Heckerman, D., and Horvitz, E. (1998). A bayesian approach to filtering junk e-mail. In *AAAI-98 Workshop on Learning for Text Categorization*.

Schapire, R. and Singer, Y. (2000). Boostexter: A boosting-based system for text categorization. *Machine Learning*, 39(2/3):135–168.

Shannon, C. E. (1948). A mathematical theory of communication. *Bell System Technical Journal.*

Shih, Y. (1999). Families of splitting criteria for classification trees. *Statistics and Computing*, 9:309–315.

Slonim, N. and Tishby, N. (1999). Agglomerative information bottleneck. *In Advances in neural information processing systems (Vol. 12). MIT Press.*

Slonim, N. and Tishby, N. (2000). Document clustering using word clusters via the information bottleneck method. *ACM SIGIR*, pages 208–215.

Sowa, J. (1999). *Knowledge Representation: Logical, Philosophical, and Computational Foundations.* PWS Publishing, New York.

Strehl, A. and Ghosh, J. (2002). A scalable approach to balanced, high-dimensional clustering of market-baskets. In *Proceedings of the 17th International Conference on High Performance Computing 2000.*

Swartout, B., Patil, R., Knight, K., and Ross, T. (1996). Toward distributed use of large-scale ontologies. In *Proceedings of the Tenth Workshop on Knowledge Acquisition for Knowledge-Based Systems, Banff, Canada.*

Taylor, M., Stoffel, K., and Hendler, J. (1997). Ontology-based induction of high level classification rules. In *SIGMOD Data Mining and Knowledge Discovery workshop.*

Towell, G. and Shavlik, J. (1994). Knowledge-based artificial neural networks. *Artificial Intelligence*, 70:119–165.

UMLS (2001). Unified medical language system. In *http://www.nlm.nih.gov/research/umls/.*

Undercoffer, J. and et al. (2004). A target centric ontology for intrusion detection: Using daml+oil to classify intrusive behaviors. In *Knowledge Engineering Review - Special Issue on Ontologies for Distributed Systems, January 2004, Cambridge University Press.*

Walker, A. (1980). On retrieval from a small version of a large database. In *Proceedings of the Sixth International Conference on Very Large Data Bases*, pages 47–54.

Wallace, C. and Boulton, D. (1968). An information measure for classification. *Computer Journal*, 11:185–194.

White, A. (1987). *Research and Development in Expert Systems III, edited by M.A. Bramer.* Cambridge University Press.

White, A. and Liu, W. (1993). Fairness of attribute selection in probabilisic induction. In Bramer, M. and Milne, R., editors, *Research and Development in Expert System IX*, pages 209–224. Cambridge Univ. Press, Cambridge, UK.

WHODD (2001). The world health organization drug dictionary. In *http://www.umc-products.com/.*

Wu, F., Zhang, J., and Honavar, V. (2005). Learning classifiers using hierarchically structured class taxonomies. In *Proceedings of the Symposium on Abstraction, Reformulation, and Approximation (SARA 2005).*

Yamazaki, T., Pazzani, M., and Merz, C. (1995). Learning hierarchies from ambiguous natural language data. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 575–583, Morgan Kaufmann.

Zhang, J., Caragea, D., and Honavar, V. (2005a). Learning ontology-aware classifiers. In *Proceedings of the 8th International Conference on Discovery Science. Springer-Verlag Lecture Notes in Computer Science. Singapore*, Vol. 3735, pages 308–321, Berlin: Springer-Verlag.

Zhang, J. and Honavar, V. (2003). Learning decision tree classifiers from attribute value taxonomies and partially specified data. In Fawcett, T. and Mishra, N., editors, *Proceedings of the Twentieth International Conference on Machine Learning*, pages 880–887, Washington, DC.

Zhang, J. and Honavar, V. (2004a). Learning concise and accurate naïve bayes classifiers from attribute value taxonomies and data. In *Proceedings of The Fourth IEEE International Conference on Data Mining*, Brighton, UK.

Zhang, J. and Honavar, V. (2004b). Learning naïve bayes classifiers from attribute value taxonomies and partially specified data. In *Proceedings of the International Conference on Intelligent System Design and Applications(ISDA 2004)*.

Zhang, J., Kang, D.-K., Silvescu, A., and Honavar, V. (2005b). Learning accurate and concise naïve bayes classifiers from attribute value taxonomies and data. In *Journal of Knowledge and Information Systems*, June, 2005.

Zhang, J., Silvescu, A., and Honavar, V. (2002). Ontology-driven induction of decision trees at multiple levels of abstraction. In *Proceedings of Symposium on Abstraction, Reformulation, and Approximation 2002. Lecture Notes in Artificial Intelligence*, Vol. 2371.